Tecnología Digital VI: Inteligencia Artificial - Trabajo Práctico 2

Federico Giorgi Gastón Loza Montaña Tomás Curzio

Análisis exploratorio de los datos

Para el análisis de los datos quisimos observar patrones de comportamiendo de los usuarios de la plataforma de e-commerce. Para ello, graficamos la frecuencia de conversión según la plataforma desde la cual está operando el usuario y la frecuencia de conversión según hora y día de la semana (figura 1).

Entendemos por cómo estaba presentada la información de la plataforma en la que se estaba realizando la visualización que tanto la categoría `android` como `ios` refieren a las aplicaciones nativas de la plataforma en esos sistemas operativos y que `mobile` refiere al uso de la plataforma desde el navegador del dispositivo móbil. Notamos que la conversión en `desktop` es superior a la demás alternativas y que dentro de las opciones de teléfonos, la mayor tasa de conversión se da en dispositivos con el OS de Apple.

De la figura 2, nos llama la atención que la tasa de conversión se mantiene bastante estable durante casi todo el día (entre las 9:00 a 23:00) con leves picos en el horario de la salida del horario laboral (18:00) y en la cena u horario de ir a descansar (22:00-23:00).

Ingeniería de atributos

Variables adicionales

A pesar de la gran cantidad de atributos con las que cuenta el dateset provisto, nos parecía interesante agregar variables adicionales que puedan aportar al modelo predictivo. Entre ellas se encuentran:

- `discount_%`: tomamos la diferencia procentual entre el precio original y precio.
- `month`, `day`, `day_of_week`, `hour`, `minute`, `second`: desagregación de la variable original `print_server_timestamp`.
- `has_warranty`: a partir la exploración de la variable `warranty` que contiene texto libre con poca estandarización, tomamos aquellas que contengan las palabras "sin" y "garantía", y que no tengan digitos numéricos. A veces se mencionan las palabras "sin" y "garantía" en el texto en otro contexto y mencionan con dígitos numéricos indicando la duración de garantía.
- `tags`: creamos para uno de los tags una columna binaria.
- `category_first` y `category_last`: a partir de la variable `full_name` que indica el nombre de completo de la categoría divido por jerarquía con "–>" decidimos tomar el primer y último nivel de esta jerarquía. Esta decisión la tomamos observando que estas dos eran las más descriptivas de la categoría y usar todos los niveles de jerarquía generaría un nivel excesivo de columna al hacer OHE sobre las mismas.
- `platform`: tomamos la última palabra del string ya que esta es la que más información aportaba.
- `title_emb{i}`: embedings en base a w2v del atributo `title`. En principio hicimos vectores de 300 para hacerle PCA y notamos que con las primeras 100 componentes se explicaba el 90%. Comparamos el AUC-ROC con las primeras 100 componentes principales y contruir los vectores con 100 dimensiones y nos dio mejores resultados lo segundo. Luego al notar que nuestro AUC de validación discrepaba bastante con el del leaderboard pública decidimos bajar las dimensiones a 50 y vimos que funcionó mejor.

Variables que no aportaban información

Además de agregar variables, decidimos eliminar variables que consideramos que no aportarían al modelo.

- Eliminamos las variables que en Kaggle decía que se debían ignorar como `benefit`.
- Decidimos eliminar `accepts_mercadopago` dado que todas las rows tenían la misma información.
- Eliminamos `main_picture` pues aún no vimos como obtener información de imágenes.
- Identificamos que `category_id` y `domain_id` contaban con la misma información, nada más que uno en forma numérica y otra de texto. Además estos atributos coinciden con `full_name`, el cuál ya usamos para crear `category_first` y `category_last`.
- `product_id` creímos que tal vez sería útil hacerle counting pero al comparar con otras versiones del modelo notamos que hacerlo no mejoraba, por lo que decidimos eliminarla.

- Los ids restantes también los eliminamos. Hacerle OHE incrementaba demasiado la cantidad de columnas y el conteo no nos había sido útil para aumentar score.

Conjunto de validación

En las clases vimos principalmente tres maneras de realizar conjuntos de validación: `Holdout Set`, `LOOCV`, `K-Fold CV`.

En nuestro modelo decidimos utilizar `Holdout Set`, con una proporción de 70% para entrenar y 30% para validación. Consideramos que, al querer un validation set similar al leaderboard de kaggle, hacer una partición del mismo tamaño podría tener sentido, y además se condice con los estandares vistos en clase.

¿Por qué decidimos hacer `Holdout Set`?

La principal respuesta es tiempo de cómputo. Leave-one-out Cross Validation lo descartamos de entrada pues era inviable con una cantidad de datos tan grande como la del dataset que estamos utilizando. K-fold CV podría ser usado con valores chicos de k, pero de igual manera cada entrenamiento lleva su tiempo, y nos parecía mas útil usar ese tiempo en pensar y aplicar posibles mejoras a nuestro modelo, con una validación que tarde menos pero igualmente sea de calidad, pues hay muchos datos para validar en un 30% de un dataset tan grande, y a su vez no duele tanto perderlos para entrenar como si podría pasar en un dataset pequeño.

Buscabamos además que el validation set no nos arroje resultados muy optimistas por sobre a lo obtenido en el leaderboard y experimentalmente logramos ese efecto de esta manera, por lo que nos pareció razonable utilizarlo de brújula para la construcción de nuestro modelo.

Modelo predictivo e hiperparámetros

Para empezar, utilizamos un `random forest` muy básico que fue nuestro punto de partida con nuestros commits iniciales, para superar el modelo básico otorgado. No nos enfocamos en optimizar los parametros ya que era simplemente un intento de mejorar un poco el modelo inicial y setear un piso, sin aún haber hecho la ingeniería de atributos mencionada.

Rápidamente cambiamos el modelo para utilizar la librería `XGBoost`, ya que se mencionó bastante en clase que de los modelos vistos era el mas potente. Empezamos probandolo con el mismo conjunto de datos que el random forest y solo fue ligeramente mejor, pero luego de aplicar la ingeniería de datos mencionada, la diferencia vista fue muy grande (XGBoost nos dio un AUC 9.3% mayor en validation). Descartamos Naive Bayes pues en clase hemos hablado de que es un buen modelo para empezar pero no suele ser lo mejor, y regresión logística, pues al tener tantas columnas si queríamos hacer polinomio 2 se complejizaba demasiado. Esto nos dejo, de los modelos vistos en clase, con XGBoost como modelo a utilizar.

Para la elección de nuestros hiperparámetros utilizamos la librería `hyperopt`, con el algoritmo de TPE, el default de la librería, que utiliza un enfoque Bayesiano. En cada paso intenta construir un modelo probabilistico de la función y elegir los parametros mas prometedores para el siguiente paso.

Lo hicimos así para un modelo inicial sobre el cual aplicamos algunas modificaciones validando tanto en validation como en el leaderboard público y una vez que tuvimos nuestro modelo final, volvimos a calibrar los parametros con otra pasada de hyperopt (no tantas iteraciones por una cuestión de cómputo), lo cual fue beneficioso para el score.

Utilizamos solo los parámetros vistos en clase, ya que esto nos permitía tener una idea mas clara de cuando podíamos estar overfitteando y cuando no al ir moviendolos. Como finalmente los parámetros de hyperopt nos daban un buen score tanto en validation como en el leaderboard público, consideramos que no estaba haciendo overfitting y no los modificamos. En el leaderboard privado nuestro score aumentó, lo que nos da pie a pensar que esa consideración era correcta.

Análisis Final

Para ver la importancia de nuestros parametros, utilizamos distintas métricas de importancia, principalmente `Gain` y `Weight`. Mientras `Gain` implica la contribucion relativa de la variable al modelo (mayor gain que otra variable significa que es mas importante), `Weight` representa la cantidad de splits que se hizo con la variable. Entendemos que el `Weight` es interesante pues si se utiliza en muchos splits tiene sentido que sea un predictor útil, pero nos basamos mas en la `Gain`, pues entendemos que por ejemplo una variable binaria muy importante solo puede generar 1 split por arbol (como es el caso de is_pdp).

Feature	weight	gain	cover	total_gain	total_cover	Importance
is_pdp	227	770.69	10377.1	174947	2355595.25	0.2932
offset	253	99.7131	4563.77	25227.4	1154634.25	0.0379
platform_desktop	210	50.7512	3302.91	10657.8	693610.25	0.0193
print_position	792	45.1744	1922.79	35778.2	1522846.25	0.0171
total_orders_item_30days	509	30.6762	2353.67	15614.2	1198020	0.0116

La tabla muestra el top 5 de variables mas importantes. Podemos ver como la posición en la página importa, determinado por las variables `offset` y `print_position`, que la gente suele hacer sus compras desde una computadora, por la variable `platform_desktop`, que los items que mas se vienen vendiendo suelen seguir esa tendencia por la variable `total_orders_items_30days` y finalmente que lógicamente es muy importante que los usuarios clickeen nuestro producto para que lo compren, por la variable `is_pdp`.

Para darle un consejo a vendedores, buscamos enfocarnos justamente en eso, en el primer paso que es lograr que el usuario clickee nuestro producto, lo cual aumenta nuestra probabilidad de conversión. Para eso, buscamos las variables que se correlacionan con `is_pdp`.

Interesantes correlaciones con is_pdp:

- price: 0.1138636035
- extended_warranty_eligible: 0.1761752217
- avg_gmv_seller_bday: 0.2081312699
- category_last_Celulares y Smartphones: 0.1965971884
- ahora-12: 0.1082403863
- platform_desktop: 0.1035357212
- free_shipping: 0.068602974
- fulfillment: 0.0677687822

Podemos ver como es muy importante `avg_gmv_seller_bday` la cantidad de ventas que tiene el vendedor, esto puede ser por un factor de confianza si lo conocemos (por ejemplo, si aparece que lo vende una marca conocida) o también porque aquellos que venden mucho, saben como operar en la página. Principalmente vemos que precios competitivos hacen la diferencia, lo cual es bastante lógico, luego hay categorías como `category_last_Celulares y Smartphones` o `platform_desktop` que tienen mas que ver con que son productos muy buscados y que en general como vimos las compras se hacen desde la computadora. Cosas que el vendedor si puede tener en cuenta son variables como `ahora-12`, `free_shipping`, `fulfillment`, que si bien le pueden generar un costo (pagar el envío o hacerlo uno mismo, dar cuotas, etc.) influyen a que el producto sea mas tenido en cuenta y clickeado, lo cual lleva a una mayor probabilidad de venta. Aun que no parezca muy lógico ya que no aparece tanto al scrollear, la variable `extended_warranty_eligible` parece tener un impacto, posiblemente por filtros o funcionamiento interno del retailer que lo hagan aparecer mas arriba (vimos como también era un factor importante), por lo que recomendaríamos ofrecer una garantía extendible de ser posible.

Teniendo en cuenta estas consideraciones, probablemente el vendedor pueda lograr un mayor porcentaje de clicks en su producto, consiguiendo así una mayor cantidad de ventas.

Posibles debilidades de este análisis son que algunas variables a tener en cuenta implican gastos adicionales y también se ve que para un vendedor nuevo puede ser dificil competir con el factor de confianza y probablemente manejo del algoritmo del retailer para mostrar los productos de los vendedores mas consolidados (con mas ventas) y no está siendo tan tenido en cuenta.

Anexo: Gráficos

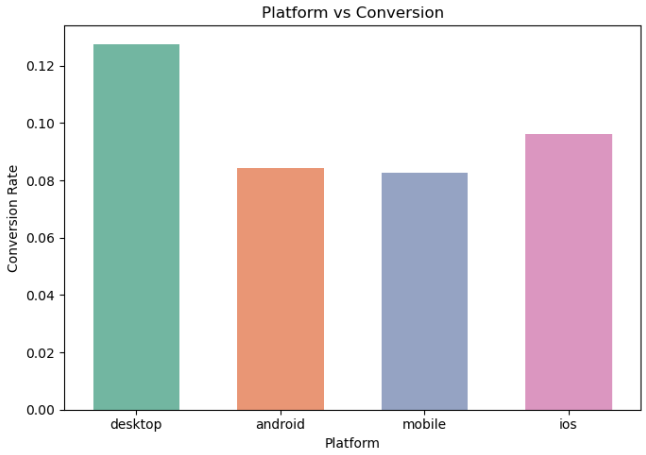

Figura 1: Histograma de conversión por plataforma

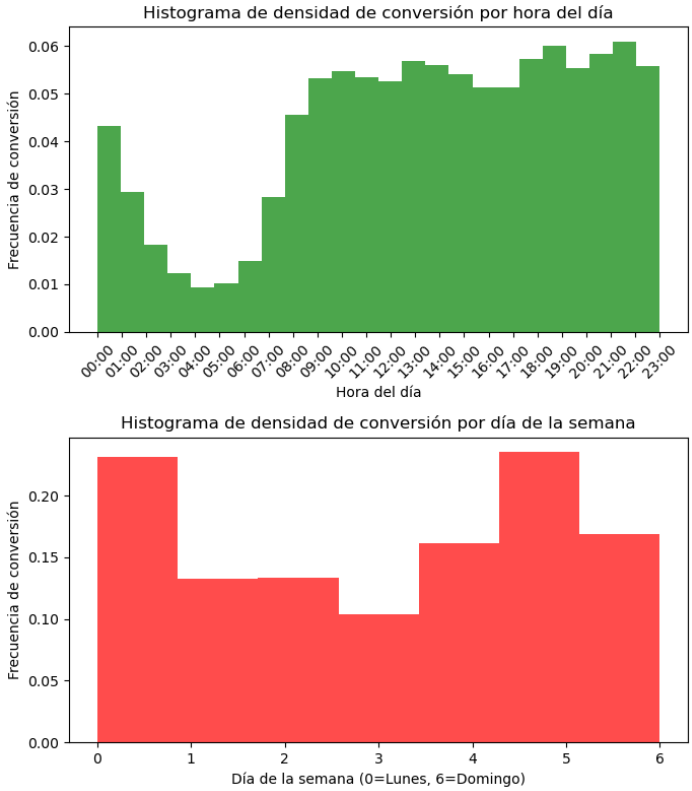

Figura 2: Histogramas de conversión por hora del día y día de la semana