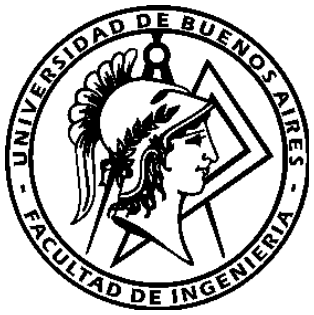

UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

Organización de Datos (75.06)

TRABAJO PRÁCTICO Nro. 2

PREDICCIÓN COMPRAS DE USUARIOS

GRUPO 11	
Ituarte, Lautaro Javier	93639
Castellanos, César	81404

Segundo Cuatrimestre 2018

1. Introducción

En el siguiente trabajo práctico buscaremos predecir si los usuarios pertenecientes al sitio web Trocafone realizarán una compra de celulares en las dos primeras semanas del mes de junio. Para eso, la empresa nos facilitó un set de datos con todos los eventos de un conjunto de usuarios desde enero hasta mayo.

Para el desarrollo de este trabajo se armó un repositorio en *github* donde se creó un notebook en jupyter para cada algoritmo a probar. El repositorio para el trabajo se encuentra en https://github.com/lituarte/Datos-TP2

2. Feature Engineering

Nuestro objetivo, como se ha aclarado previamente, es predecir si los usuarios realizarán compras en el sitio web. Para esto necesitamos tener datos expresados por usuario en lugar de por evento, como los teníamos inicialmente.

Como primera actividad realizada se expandió la columna de eventos para poder obtener una columna para cada evento posible. Este cambio nos ayudará a contar la cantidad de cada uno de los eventos realizados por el usuario una vez agrupemos los datos por usuario.

Una vez hecho esto, utilizamos la columna de *timestamp* para obtener datos sobre meses y semanas anterior, los cuales utilizamos particularmente para determinar cuando ocurrieron los eventos. La columna “mes_pasado_0” es True si el evento ocurrió en el mes actual (Es decir que el “mes actual” es mayo, ya que estamos desde la perspectiva de querer predecir junio y el último mes del cual tenemos información es mayo), la columna “mes_pasado_1” es True si el evento ocurrió el mes anterior y así sucesivamente.

Una vez que realizamos estos cambios agrupamos por usuario y filtrando algunas columnas nos quedamos con un dataframe con las siguientes columnas:

- '**person**': contiene el identificador del usuario
- '**viewed_product**', '**brand_listing**', '**visited_site**', '**generic_listing**', '**searched_products**', '**search_engine_hit**', '**checkout**', '**conversion**' y '**lead**': contienen la cantidad de veces que

el usuario participó de alguno de esos eventos. Obtuvimos mejores resultados al filtrar los usuarios con números demasiado altos.
- '**mes_pasado_0**', '**mes_pasado_1**', '**mes_pasado_2**' y '**mes_pasado_3**': cantidad de eventos producidos por el usuario en su mes correspondiente.
- '**semana_pasada_0**', '**semana_pasada_1**', '**semana_pasada_2**', '**semana_pasada_3**' y '**semana_pasada_4**': cantidad de eventos producidos por el usuario en la semana correspondiente.

Estos son los features base con los que trabajamos, para cada algoritmo particular fuimos agregando o quitando features para obtener mejores resultados. Estos cambios en features según algoritmo serán descritos en la sección del algoritmo correspondiente.

Es importante destacar que tanto los features, como los filtros que se han aplicado a los datos disponibles fueron pensados en base a los resultados obtenidos en el TP 1; como la decisión de dejar las marcas de celulares más populares (Samsung y iPhone), el filtrado de usuarios con demasiados eventos o la decisión de considerar solo 3 categorías para el estado del producto (“Excelente”, “Bom” y “Moito Bom”).

3. Algoritmos utilizados

3.1. Logistic Regression

Se utilizó este algoritmo inicialmente junto con Random Forest para poder tener referencias sobre con cuáles algoritmos se debía de realizar el trabajo.

3.1.1. Evolución de los features e hiperparámetros a través del tiempo

Para este algoritmo se agregaron los siguientes features a los “features base” mencionados anteriormente:

- **'Samsung'**, **'iPhone'**, **'LG'**, **'Motorola'**, **'iPad'**, **'Sony'**, **'Lenovo'**, **'Quantum'**, **'Asus'**, **'Xiaomi'** y **'Outros'**: cantidad de veces que el usuario provocó un evento relacionado a esos modelos (basado en la columna “model”)
- **'dia_pasado_0'**, **'dia_pasado_1'**, **'dia_pasado_2'**, **'dia_pasado_3'**, **'dia_pasado_4'**, **'dia_pasado_5'** y **'dia_pasado_6'**: cantidad de eventos producidos en los días correspondientes.

3.1.2. Resultados

Para este algoritmo se probó con los hiperparametros que vienen por defecto y, a pesar de obtener un score de 0.94, obtuvimos un AUC local de 0.71 y un resultado de Kaggle de 0.68. Como estos resultados no eran muy prometedores, no profundizamos en su uso.

3.2. XGBOOST

A priori este era el algoritmo del que mejor resultados esperábamos, por lo que se focalizó mucho esfuerzo en su funcionamiento, variando features e hiperparametros.

3.2.1. Evolución de los features e hiperparámetros a través del tiempo

Inicialmente se probó con los "features base" y se obtuvo un prometedor resultado de 0.84129 en Kaggle en el primer intento. A partir de entonces se fueron agregando y quitando features para obtener el mejor resultado posible.

- **'Samsung'**, **'iPhone'**, **'LG'**, **'Motorola'**, **'iPad'**, **'Sony'**, **'Lenovo'**, **'Quantum'**, **'Asus'**, **'Xiaomi'** y **'Outros'**: se probó agregando features que contabilizaran la cantidad de veces que un usuario disparó un evento relacionado a las marcas que comercializa Trocafone. Aunque la mayoría de estos features no aportaba al modelo, y daban muy bajo en el gráfico de feature importance. Sin embargo, las dos columnas más prometedoras son las de "Samsung" e "iPhone" ya que son las marcas más populares dentro del sitio web.
- **'busquedas_exactas'** y **'terminos_buscados'**: se exploraron algunos features relacionados a las búsquedas realizadas por el usuario; "busquedas_exactas" es la cantidad de búsquedas que coinciden exactamente con un modelo de celular ofrecido por Trocafone, mientras que "terminos_buscados" es la cantidad de términos que buscó el usuario. Sin embargo, ninguna de estas dos features dio buenos resultados para el objetivo del trabajo.
- **'Bom'**, **'Muito_Bom'** y **'Excelente'**: se agregaron features para cada uno de los estados posibles para los productos, produciendo buenos resultados.
- **'dias_ultimo_evento'** y **'dias_ultima_compra'**: estas dos features saltaron inmediatamente a los primeros puestos de *feature importance* cuando fueron agregadas: "dias_ultimo_evento" es la cantidad de días que pasaron desde el último evento registrado por el usuario, mientras que "dias_ultima_compra" es la cantidad de días desde que el usuario realizó una compra (en caso de que no haya realizado ninguna compra se contabiliza el primero de enero como última compra).
- **'dia_pasado_0'**, **'dia_pasado_1'**, **'dia_pasado_2'**, **'dia_pasado_3'**, **'dia_pasado_4'**, **'dia_pasado_5'** y **'dia_pasado_6'**: agregamos features para la cantidad de eventos según el día para la última semana, pero no trajo buenos resultados.
- **Features de meses anteriores**: si bien obtuvimos un buen resultado inicial con las features base y lo pudimos mejorar un poco con algunas features nuevas, hay un aspecto del modelo que creemos puede ser problemático: las features históricas. Llamamos features históricas a las que incluyen eventos que ocurrieron desde "el principio de los tiempos", en nuestro caso tenemos solamente 5

meses de data por lo que no sería tan grave pero aún así, ¿importa lo que hizo un usuario en enero para determinar si va a comprar ahora?, ¿y lo que hizo hace 3 años? Es por esto que cambiamos un poco el modelo y reemplazamos las features históricas por su equivalente para el último mes, obteniendo resultados incluso un poco mejores en comparación al modelo anterior.

Tras muchas pruebas nos quedamos con:

- **'person'**: identificador del usuario.
- **'conversion'**: cantidad de conversiones históricas del usuario.
- **'viewed_product_mes_pasado_0'**, **'brand_listing_mes_pasado_0'**, **'visited_site_mes_pasado_0'**, **'ad_campaign_hit_mes_pasado_0'**, **'generic_listing_mes_pasado_0'**, **'searched_products_mes_pasado_0'**, **'search_engine_hit_mes_pasado_0'**, **'checkout_mes_pasado_0'** y **'conversion_mes_pasado_0'**: cantidad de eventos en el último mes por cada evento.
- **'mes_pasado_0'** y **'mes_pasado_1'**: cantidad total de eventos en los ultimos 2 meses.
- **'Semana_pasada_0'**, **'semana_pasada_1'**, **'semana_pasada_2'**, **'semana_pasada_3'** y **'semana_pasada_4'**: cantidad total de eventos en las ultimas 5 semanas.
- **'Dias_ultimo_evento'**: cantidad de dias desde que el usuario realizo su ultimo evento.
- **'Dias_ultima_compra'**: cantidad de dias desde que el usuario realizo una compra.
- **'Bom_mes_pasado_0'**, **'Muito_Bom_mes_pasado_0'** y **'Excelente_mes_pasado_0'**: cantidad de eventos relacionados con cada estado posible de los productos.
- **'Samsung_mes_pasado_0'** y **'iPhone_mes_pasado_0'**: cantidad de eventos relacionados con las marcas más populares del sitio.

A medida iba cambiando el modelo también fueron cambiando los hiperparametros, el método más utilizado para optimizarlos en este algoritmo fue grid search con cross validation, aunque también se utilizó el randomized grid search.

3.2.2. Expansión del set de datos

Con la transición de un modelo con features "históricas" a uno con features reducidas a los últimos dos meses surgió un nuevo interrogante: ¿Qué hacemos con los datos de los demás meses que actualmente no utilizamos? La respuesta que encontramos fue expandir el set de datos.

Para la realización del trabajo práctico fuimos provistos por los labels para un grupo reducido de usuarios para el mes de junio, el set de datos en sí contiene información sobre la compra de productos por parte de los usuarios, la cual sigue siendo utilizable a los fines del trabajo. Para el mes de mayo por ejemplo, tomamos las compras realizadas las primeras dos semanas para armar los labels y los eventos de abril y marzo para armar los features.

Lo que hicimos fue cambiar un poco el enfoque del entrenamiento del modelo de “dados los datos de todo el año predecir si un usuario realizará una compra en las próximas dos semanas” a “dados los datos de los últimos dos meses predecir si un usuario realizará una compra en las dos semanas siguientes”.

3.2.3. Resultados

Para el modelo entrenado exclusivamente con los labels otorgados por la cátedra obtuvimos localmente un AUC de 0.8723510506668903 y un RMSE de 0.21087221488665722, mientras que en kaggle tuvimos un resultado de 0.85663.

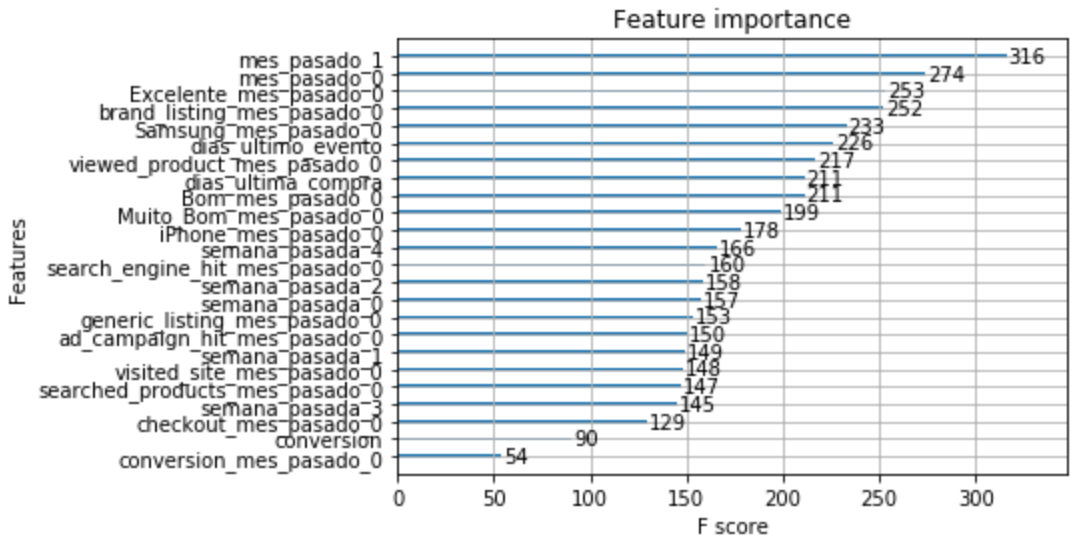

Gráfico de importancia de features para set de datos sin expandir

Para el modelo entrenado con el set de datos expandido hasta abril obtuvimos localmente un AUC de 0.8392069286911953 y un RMSE de 0.18754107297777106, mientras que en kaggle obtuvimos un resultado de 0.85797.

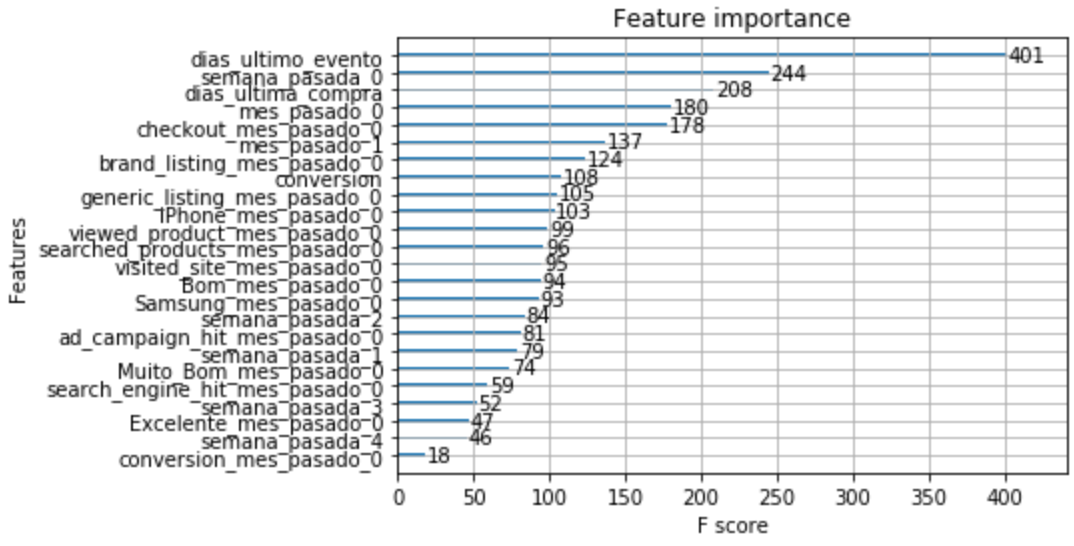

Gráfico de importancia de features para set expandido hasta abril

También entrenamos un modelo con los datos expandidos hasta marzo y obtuvimos un AUC local de 0.8457985291321034, un RMSE de 0.18165574682854405 y un puntaje de Kaggle de 0.85822 mejorando al modelo de abril en todas las métricas.

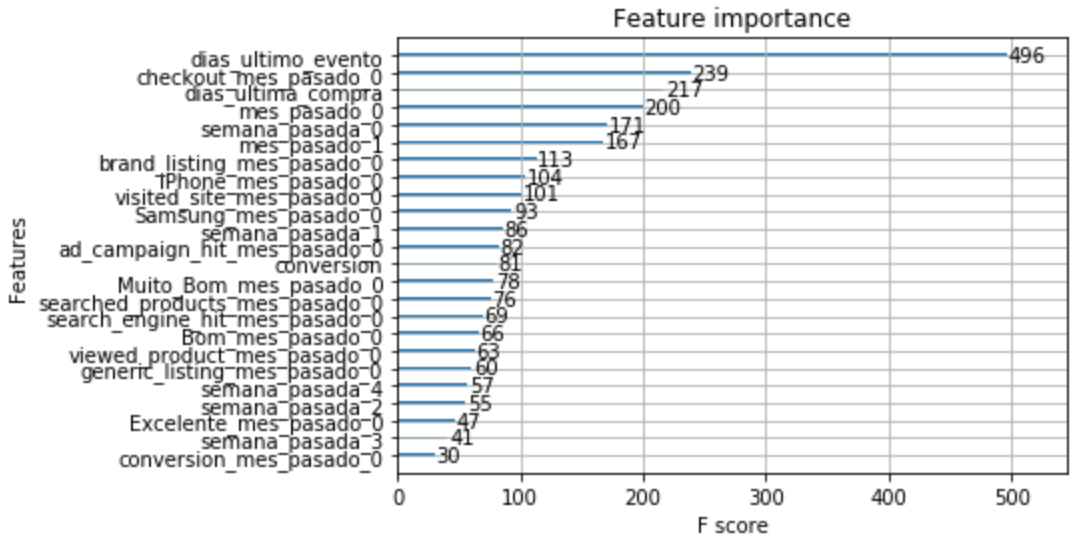

Gráfico de importancia de features para set expandido hasta marzo

A pesar de utilizar los mismos features, se notan diferencias importantes en las features más importantes para cada modelo. En el caso del modelo sin expandir, las features más importantes son la cantidad de eventos del mes anterior y del actual que ocupan el puesto 4 y 6 para el modelo expandido hasta abril, que considera como features de mayor importancia los días que pasaron desde el último evento, la cantidad de eventos en la última semana y la cantidad de días desde la última compra. El modelo expandido hasta marzo profundiza la importancia de los días desde el último evento y tiene algunos cambios en el orden de importancia.

Nota: por cuestiones de tamaño no pudimos incluir los gráficos de los árboles producidos por cada modelo pero dichos gráficos se encuentran en el repositorio.

3.3. RANDOM FOREST

3.2.1. Evolución de los features e hiperparámetros a través del tiempo

Habiendo trabajado con los features utilizados en XGBOOST (detallados en el punto 3.2.1) se realizaron pruebas con este algoritmo como lo fue también con Logistic Regression.
Los primeros resultados, con la configuración básica de los hiperparametros (con 100, 200, 500 y 1000 estimadores) resultaban cercanos a los obtenidos inicialmente en XGBOOST. Se decidió hacer submits en Kaggle con distintas configuraciones de estimadores. El primer resultado nos dio 0.81826 y luego de varias pruebas, con 1000 estimadores se obtuvo un score de 0.82570 por lo que se le dedicó tiempo de trabajo para intentar mejorarlo y para tener una alternativa de prueba.

3.3.2. Resultados

Luego de los primeros submit, se intentó mejorar los resultados por lo que se optimizaron sus hiperparametros utilizando RandomizedSearchCV con el set de datos hasta abril, lo cual dio una configuración que registró un AUC local de 0.821419 y obteniendo en Kaggle un resultado de 0.86315. Luego se hicieron distintos tipos de pruebas, trabajando con los features pero no logramos mejorar este resultado.

4. Conclusiones

De los algoritmos probados para el set de datos con los features básicos obtuvimos los resultados esperados: la logistic regression fue la que peor resultado dio y XGBoost, el algoritmo del que más esperabamos por su fama de ganar competiciones, arrojó el mejor resultado. Lo que no esperábamos era que luego de realizar algunos cambios en los features y optimizar hiperparametros, el mejor submit fuera el obtenido con Random Forest.

Pudimos comprobar las mejoras tanto en AUC como en RMSE al extender nuestro set de entrenamiento hasta el mes de marzo, creemos que la disponibilidad de más información hace que el modelo generalice mejor.

Si bien la optimización de hiperparametros juega su rol, encontramos que la feature engineering es la que más diferencia ha marcado a lo largo del desarrollo del TP.