FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2

[7506/9558] Organización de Datos
4
Curso 1
Primer cuatrimestre de 2019

Alumno 1:	BONFIL, Lucas Javier	101076
Alumno 2:	SCHISCHLO, Franco Daniel	100615
Alumno 3:	DIAZ MULLIGAN, Gonzalo	101590

https://github.com/luc662/Organizacion-de-datos

Índice

1. Introducción

El presente informe reúne la información del segundo trabajo práctico, realizado por los alumnos de la materia Organización de Datos [7506/9558] con los sets de datos proporcionados por la empresa Jampp. Este informe es un resumen de la predicción del tiempo estimado para que un cierto dispositivo aparezca en las subastas para una ventana de 3 días, de la cual no tenemos información, como también del tiempo hasta que ese mismo dispositivo convierta. Para realizar esto, utilizamos la información conseguida en el análisis exploratorio previo, el cual esta en el repositorio de github previamente mencionado. Por cuestiones de seguridad, dicho repositorio es privado, para poder acceder a dicho repositorio, se pide mandar un mail a lucas_bonfil@hotmail.com

2. Armado del set de datos

En la siguiente sección se describirá las formas de idear los set de datos realizados por los alumnos.
Dado el problema a analizar, debemos predecir el comportamiento de los dispositivos. Por esto se deben agrupar los Dataframes ya sea por el ref_hash, o device_id, según corresponda.
Otro problema a solucionar es la forma en relacionar las features con los labels correspondientes. Ya que necesitamos predecir labels de los 3 días siguientes a los últimos datos que poseemos, debemos relacionar en el set de entrenamiento features con labels de los 3 días siguientes.
En el csv de auctions solo cuenta con información de 9 días. La consigna solo nos permite utilizar datos de los 3 días anteriores para la predicción final del target, por lo cual decidimos hacer features que utilicen solo ventanas de 3 días. Los features de training solo son las ventanas de 3 días de las cuales podríamos calcular el st y sc de los siguientes 3 días, estas son:
Features ventana 1 ->Labels ventana 4
Features ventana 2 ->Labels ventana 5
Features ventana 3 ->Labels ventana 6
Features ventana 4 ->Labels ventana 7
Siendo la ventana 1, la ventana de tiempo desde el primer día de los 9, hasta el tercero.

También, trabajamos con un modelo reducido, en en cual usamos solo dos ventanas disjuntas, donde quedaba un set de train formado por: Features ventana 1 ->Labels ventana 4
Features ventana 4 ->Labels ventana 7
Esta ventana de tiempo al tener menos datos era más rápido que el modelo completo, pero su predicción solía ser peor, por otro lado, este set de datos resultaba practico a la hora de modificar hiperparámetros, ya que reducía el tiempo de ejecución.

Luego, como se debe predecir para la ventana entre el 27 y el 30, la ventana de tiempo utiliza los features correspondiente a la ventana 7, los últimos 3 días.

Este modelo nos permite generar un set de entrenamiento mucho mas grande que el set que contamos para la predicción del target, lo que mejora el entrenamiento, pero aumenta el tiempo que se tarda para realizarlo. Por lo tanto, lo utilizamos para el cálculo del target final, pero para realizar pruebas de distintos modelos, como KNN, o como pruebas de hiperparámetros, utilizamos una versión reducida de este set, ya sea un sample, o la utilización de una sola ventana.

Otro problema del set de training es que contamos con mucha información de dispositivos que no convierten, o que no aparecen en subastas. Teniendo en cuenta que tenemos que predecir el tiempo de conversiones de para dispositivos que convierten, esto entrena de forma incorrecta al predictor, ya que para estos valores debemos rellenar el NaN con el tiempo máximo de la ventana, es decir 3 días pasados a segundos, y este valor es falso. Para solucionarlo filtramos una porción de los valores que no tienen un valor de sc o st.

Para joinear los features provenientes de distintos scv, ya que no cuentan todos con información de los mismos dispositivos, hay que joinearlos. Decidimos realizar outer joins, ya que si realizamos

inner joins el set de datos queda con menos de 20 registros.

Con el set completo, algunos features se pueden llenar con 0, como la suma de repeticiones, o los valores de one hot encoding. Aunque a simple vista parece necesario realizarlo, los resultados de las predicciones fueron los mismos al realizar este cambio, por lo tanto no lo realizamos. Los features de tiempos, se rellenaron con el tiempo máximo de la ventana. Este cambio si tuvo resultados visibles.

3. Selección de features

A continuación vamos a mostrar cómo obtuvimos los datos necesarios para realizar la predicción del trabajo.
Se seleccionaron de features de 2 maneras distintas, una .ªutomática", y una manual. La primera fue, para cada csv, separamos el día y la hora en dos columnas distintas, luego, para cada columna numérica, obteníamos los siguientes features ['sum', 'mean', 'std', 'min', max'].
Para las variables categóricas, o las columnas hasheadas con baja entropía, realizamos One Hot encoding, pero dado que algunas columnas tienen muchos valores posible realizábamos las operaciones solamente con los N principales valores generales. Esto resultaba con varios registros con columnas llenas de 0s y 1s representando si ese registro tiene el valor que especifica esa columna o no, las cuales decidimos aplicar de nuevo las operaciones de ['sum', 'mean', 'std', 'min', 'max'], donde; 'sum' sirve para representar la cantidad total de entradas con ese dato, 'min' y 'max' pueden servir para indicar si es el único valor para ese dispositivo, y por ultimo, 'mean' y 'std' nos permite ver la proporción entre 1's y 0's que obtuvimos.

Esto significo en una cantidad gigantesca de columnas, cercana a 1000, por lo tanto realizamos diferentes métodos de poda para seleccionar los mas útiles. Probamos realizar con una muestra reducida del set de datos la selección de los 50 features mas importantes para Sc tanto como para St, utilizando XGBoost y random forest, y 2 podas manuales de las features.
La poda de XGBoost fue mejor que la de random forest, y también fue mejor que la poda manual realizada por un alumno del grupo. Pero la selección manual realizada por otro alumno del grupo fue la que tuvo mejores resultados.
La poda manual con malos resultados consiste en solamente la suma y el promedio de las columnas de One hot encoding, la suma para las columnas de repeticiones y el mínimo, máximo, promedio y desviación estándar de las variables numéricas.
La poda manual con mejores resultados consiste solo en la información proveniente de columnas numéricas, sin utilizar el one hot encoding.

Otras features utilizados fueron el valor de St y de Sc para la ventana actual, que resulto ser un feature muy importante. Ya que es la misma variable que queremos predecir pero en la ventana anterior.

3.1. Features desechados

Muchos features que se crearon de forma manual fueron eliminados. Por lo general, estas incluían de alguna forma las variables categóricas o booleanas, y dado que estamos trabajando con una predicción numérica, y uno de los algoritmos a usar era XGBoost, no resulta sorprendente que sean desechadas. Algunos ejemplos son:

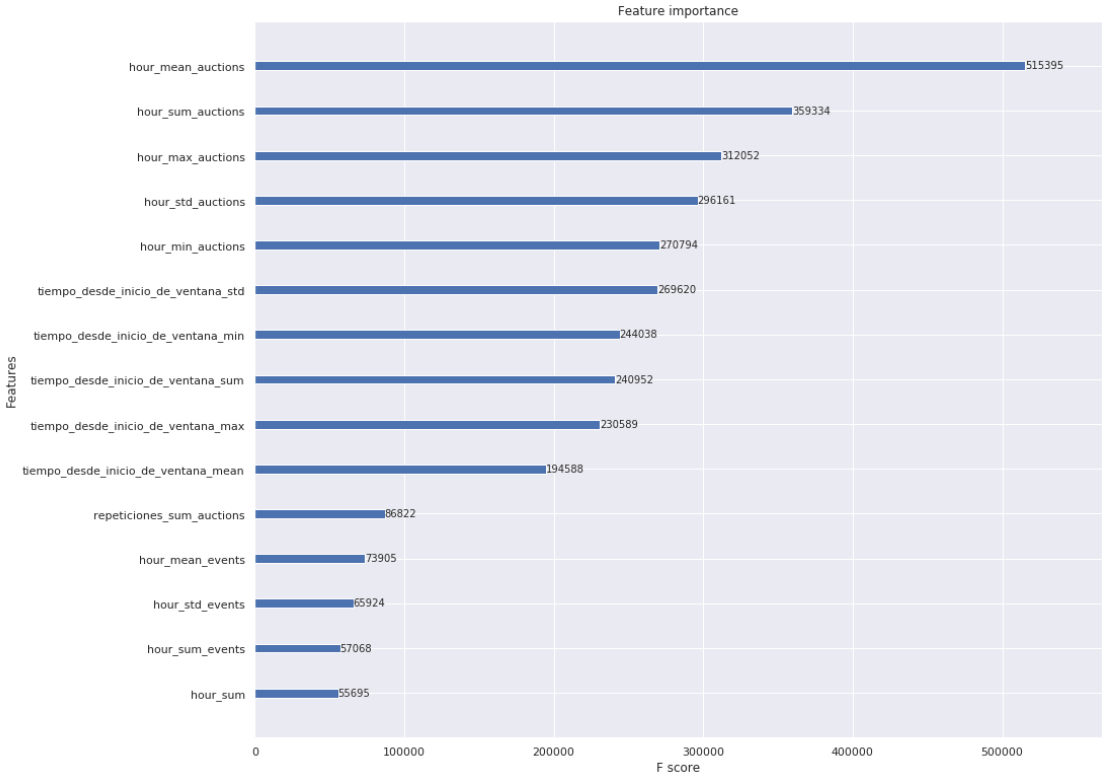
- Expansión de booleanos con One Hot Encoding: esta feature se logró realizar de forma automática en la sub-sección anterior. Pero la mejor predicción realizada no contaba con ningún tipo de one hot encoding. Esto nos indica que no existe relación entre el valor de las variables categóricas y las variables que necesitamos predecir
- Categórico mas repetido en un momento dado: esta feature consistía en contar los elementos categóricos que realizó una persona, y luego buscar a que categoría corresponde, un ejemplo

está en events, en el cual intentamos ver en que ciudad ingreso una persona mas veces. Se termino descartando porque era una variable puramente categórica, y el algoritmo principal a usar, XGBoost, no los soporta.

- función de pandas Get_dummies: Esta fue la forma original en la que realizamos el one hot encoding pero generaba una cantidad de columnas que no se podía procesar.
- ref_hash: Inicialmente contábamos con features calculados a partir del ref_hash o el device_id, que fueron eliminados porque causaban overfitting
- time_diff: Una feature que probamos utilizar era calcular el tiempo entre los registros de un mismo dispositivo calculando el diff del tiempo y con estos tiempos calcular el promedio, desviación estandard, mínimo y máximo. No los utilizamos para la predicción final ya que no la mejoraban. Es posible que no mejore la estimación ya que contando con la información de las horas y el numero de registros se hace redundante para XGBoost

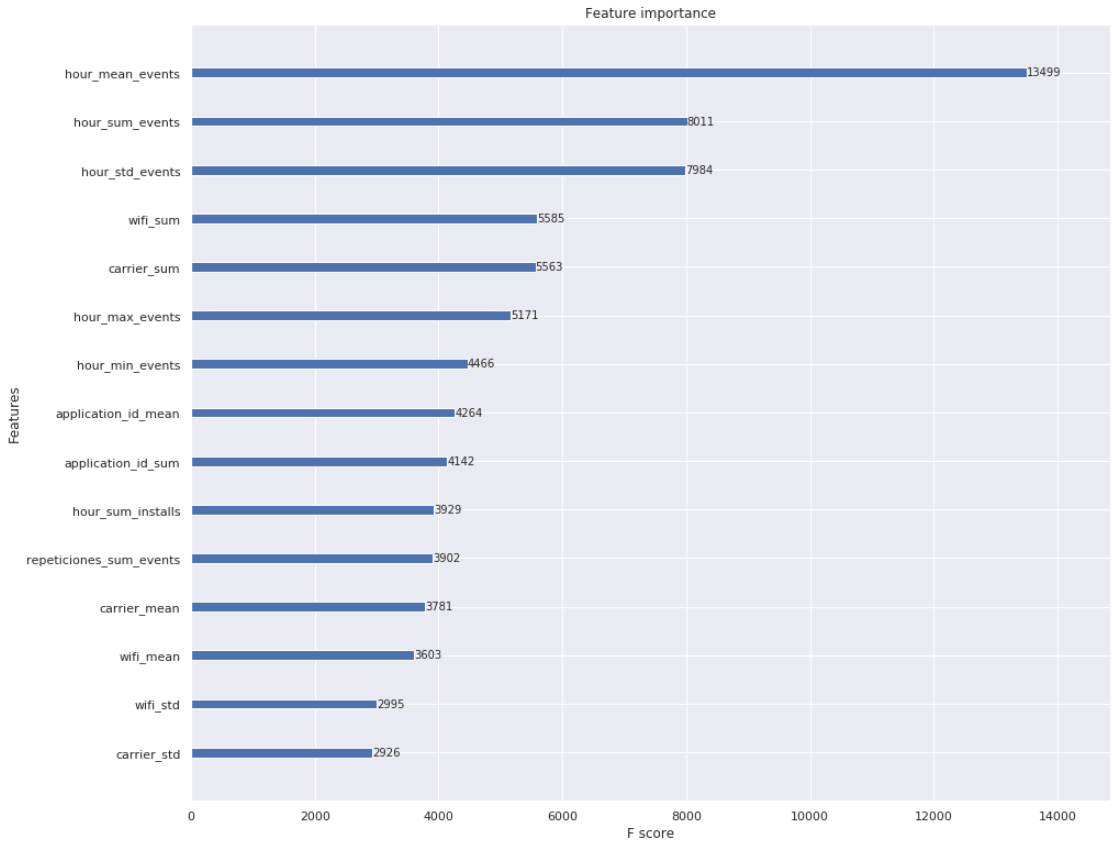
3.2. Importancia de los features

En la predicción final el gráfico de la importancia de los features, para los 15 features mas importantes de St



Como podemos ver la mayoría de los features importantes están relacionados con el tiempo de los registros en el csv de subastas, ya sea por la hora o por el valor de el St en esta ventana. Como también la cantidad de registros para este dispositivo en subastas. Y también resultan importantes los features de la hora a la que hay eventos.

Para los 15 features mas importantes de Sc



Los features mas importantes para esta prediccion resultaron ser la informacion de las horas en las que se realizan eventos, y en menor manera la hora de instalaciones. Otras features importantes son las provenientes del wifi, tanto la cantidad de registros con wifi, como el índice y la desviación estandard. También es importante el id de la aplicacion, y el numero de registros en events. La informacion proveniente del carrier_id

4. Algoritmos usados

Para realizar las predicciones usamos los siguientes algoritmos:

4.1. XGBoost

Este algoritmo es uno de los mas óptimos en cuanto a tiempo de ejecución y resultado, con los hiperparámetros correctos. Fue el algoritmo que decidimos utilizar pero al realizar poda de features con este algoritmo los resultados no fueron óptimos, y los features seleccionados por este algoritmo no suelen ser los mejores para otros algoritmos.
Al realizar la predicción con los features sin podarlos anteriormente resultó ser relativamente lento y impreciso. Otro problema que nos genera el uso de features sin podarlo, fue el hecho de que la búsqueda de hiperparametros óptimos demoraba mucho y los resultados estaban overfitteados. Dentro de los hiperparámetros que mejoraron el RSME, se encuentran max_depth, learning_rate y n_estimators siendo max_depth la altura máxima que alcanza el árbol interno de XGBoost, learning_rate el índice de aprendizaje y n_estimators cuántos arboles se realizan, por lo cual, es lógico que a mayor valor, mayor tiempo de proceso. Una forma de encontrar buenos valores iniciales fue simplemente correr la predicción varias veces con resultados distintos y ver si bajaban o no, luego a la hora de automatizar se usa un solo valor cercano al predicho manualmente, y unos

cuantos valores distintos, aunque por desgracia, no pudimos aprovechar todo el potencial de estos estimadores. Una ventaja de este algoritmo, es que puede aprender relativamente rápido, por lo que con un set de datos chico, puede obtener resultados bastante buenos. Si bien tiene un Score de features, no tiene sentido en ser utilizado en otros algoritmos, ya que, XGBoost resulta ser óptimo al tener muchas features chicas, mientras que muchos otros algoritmos para realizar las predicciones, necesitan valores mas complejos. XGBoost tiene la opción de calcular distintas funciones. Si bien puede calcular survival análisis, esta predicción sola resultó ser muy poco eficiente, ya que siempre quedaban valores muy bajos. Por otro lado, la regresión lineal solía dar los mejores resultados.

4.2. Múltiples XGBoost

Otro modelo de predicción que se intentó usar fue entrenar múltiples modelos de XGBoost. Dado que con ciertos hiperparámetros pueden dar resultados por debajo de los esperados, mientras que otros dan resultados por encima de los esperados, por lo que un modelo posible es generar múltiples predictores, y calcular el promedio. Con suficientes modelos la predicción puede llegar a mejorar. Algo a tomar en cuenta es que el pivoteo de hiperparámetros se vuelve más difícil. Además de utilizar el booster con los hiperparametros óptimos, generamos 4 booster con diferentes valores para la profundidad máxima, y ritmo de aprendizaje. Cada booster con una diferente muestra del 50 % del set de aprendizaje Este Modelo no presento mejoría con respecto a XGBoost simple, por lo que decidimos intentar un cambio de enfoque. El algoritmo pudo no ser muy efectivo ya que no había muchas diferencias en como se entrenaba, ya que se usaban el mismo set de train para los múltiples XGBoost.

4.3. Random Forest

El algoritmo de Random Forest resulto ser útil para la poda de features, aunque no fueron los features que se utilizaron en la mejor predicción, nos ayudaron a entender cuales eran los mas importantes para facilitar la poda manual. El score que estima este algoritmo para los features no se ve alterado por algoritmos internos, como si puede ocurrir con XGBoost, y, ya que funciona con probabilidades, su score es el más parecido existente a el verdadero score máximo. Se obtiene de forma rápida, por otro lado, sus predicciones no son tan buenas como la de otros algoritmos. Aun así, cuenta con la ventaja de utilizar pocos hiperparámetros, lo cual lo hace fácil de optimizar.

4.4. KNN

Este algoritmo es muy interesante, pero desde el principio veíamos difícil que genere buenos resultados. Uno de las mayores problemas que enfrenta este algoritmo; es cómo tratar a los NaN. Mientras que XGBoost permite trabajar con datos NaN, KNN no, por lo que tendríamos que rellenar valores NaN y eliminar las columnas que no tiene sentido rellenarlas porque significaría inventar datos. Una vez que se realizara esa limpieza de datos, nos quedo un set de datos con muchas menos features. Resultó ser el que peor resultados generó, no solo en predicciones, sino también en tiempos de ejecución, para realizar la predicción con este algoritmo en un periodo razonable de tiempo es necesario usar una pequeña muestra del set de datos, lo que aumentaba aun mas el error obtenido.

5. Conclusiones

5.1. Sobre XGBoost

XGBoost en este problema resultó ser un buen algoritmo, pero hay que utilizarlo con cuidado. Con respecto a los hiperparámetros, pudimos notar unas cuantas cosas.

- La primera, es que son estos valores pueden alterar mucho el tiempo de entrenamiento, pasando de menos de un minuto hasta media hora.
- Si bien un pivote automático es lo ideal(por ejemplo utilizando cross validation), nosotros no notamos un cambio muy significativo en la predicción, con respecto a los valores que obtuvimos por el pivoteo manual.
- cuánto más features tiene un set de train, más útil resulta max_depth,a costa de no escalar tan bien en los casos con menos features.Aún así, hay que tener cuidado con max_depth, ya que si se tiene un set de datos muy grande, puede generar overfitting, ya que puede usar muchos datos.
- El hiper-parámetro que más influyó realizó en nuestra predicción fue n_estimator, el cual pudo verse potenciado por la gran cantidad de valoren numéricos generados.

5.2. Sobre el modelo

El mayor problema que hubo ocurrió a la hora de calcular el tiempo hasta conversión, esto se debe a que había muy pocas personas que participaban en dos ventanas de tiempos consecutivas. Nuestra solución a ese problema fue agarrar a todas las personas que realizaron una conversión, y una fracción de las personas que no, así las predicciones ya no van a tender tanto hacia el final de la ventana.

5.3. Sobre La importancia de features

Tanto para St y Sc las features más importantes están relacionadas con las horas en las que la persona entra. Una posible cause para esto, puede ser que las personas suelen instalar aplicaciones o abrir que generan subastas en horarios similares. En el caso de Sc, la hora de los events resulta ser mas importante que la del installs, simplemente porque hay muchos mas registros de events que de installs. Otra Feature en Sc que resulta curiosa esta relacionada con la conección WiFi, lo cual podemos estimar que la cantidad de entradas con conección (o sin ella) resulta ser importante en el comportamiento de las instalaciones. Sobre St, podemos concluir que el comportamiento de las personas en las subastas, no depende mucho de los datos del resto de los csv, sino que depende en gran medida del resto del comportamiento de los auctions.