FACULTAD DE INGENIERIA
Universidad de Buenos Aires

# Trabajo Práctico Nro. 2
# Machine Learning

## Competencia Jampp

[7542] Organización de Datos
Primer cuatrimestre de 2019


| **Alumno:** | **Padrón** |
|---|---|
| ROTILI, German | 99722 |
| ALVAREZ JULIA, Santiago | 99522 |
| MONTES, Gaston | 89397 |

# Índice

# 1. Introducción

*Jampp* es una plataforma de **Performance Marketing**, pues su negocio está basado en el uso de tecnología predictiva para minimizar costos y maximizar las ganancias del **Advertiser**, en 2 tipos de campañas de marketing: **User Acquisition** y **Retargeting**. En el presente trabajo se realiza feature engineering de los datos que nos facilitó dicha plataforma, basados en eventos realizados por los usuarios que utilizan las aplicaciones partner de *Jampp* y la información que le proveen los Ad Exchanges, para entrenar un algoritmo de machine learning con el objetivo de predecir cuando un usuario va a realizar un install y cuando va a aparecer en una subasta, para así poder conseguir que se le atribuya la conversión del usuario a *Jampp*.

En la competencia de *Kaggle* se tomará como evaluación el Root Mean Squared Error (RMSE), cuanto menor sea este, mejor son nuestras predicciones.

# 2. Análisis Previo

En la primera entrega del trabajo práctico desarrollamos un análisis exploratorio que nos permitió encontrar posibles features a la hora de predecir lo descrito en *Introducción*. Como explicaremos más adelante en este informe, algunos fueron de vital importancia y otros no tanto. En la sección *Más Información* se encuentra un link al repositorio de github donde está el código utilizado en dicho análisis.

# 3. Feature Engineering

Para poder entrenar al algoritmo de machine learning se realizo feature engineering sobre los csv provistos por *Jampp*, para conseguir features ayuden a mejorar las predicciones. Para realizar las predicciones se armo un feature target que mide el tiempo hasta uno de los eventos esperados en ventanas de 3 días, ubicando el tiempo 0 en el primero de los tres días.

## 3.1. Installs

Para las instalaciones, el feature engineering se planteo con el objetivo de desarrollar la información provista por el csv de installs en features que sean impactantes para la capacidad del algoritmo de machine learning en realizar las predicciones.

### 3.1.1. Cantidad de Installs

Cada entrada del csv de Installs, tiene información de un install realizado por un usuario, usando esto se calculo la cantidad de installs que realizo cada usuario.

### 3.1.2. Cantidad de Instalaciones con wifi

Para cada instalación se tiene información de si esta se realizo con acceso a wifi o no. Se calculo la cantidad de instalaciones que se realizaron teniendo wifi, sabiendo la fracción de instalaciones con wifi, podemos saber los hábitos del usuario con respecto a las mismas.

#### 3.1.3. Marca de celular del user

De cada instalación se sabe cual era la marca de celular que se uso, con esta información, se puede realizar un perfil del usuario y ubicarlo dentro de un grupo que puede compartir hábitos de uso. Para esto se busco la marca de celular de cada usuario y se agrego como feature.

#### 3.1.4. Modelo de celular del user

De cada instalación se sabe cual era el modelo de celular que se uso, con esta información, se puede realizar un perfil del usuario y ubicarlo dentro de un grupo que puede compartir hábitos de uso. Para esto se busco el modelo de celular de cada usuario y se agrego como feature.

#### 3.1.5. Id de la aplicación instalada

Cada instalación tiene información de que aplicación se instalo, a partir de esto generamos un feature que lo muestra el id de la aplicación instalada.

#### 3.1.6. Cantidad de aplicaciones distintas instaladas por un user

Además se puede calcular la cantidad de aplicaciones distintas que instalo un usuario.

#### 3.1.7. Device Language

Cada instalación viene con información del lenguaje del dispositivo usado para la instalación, parecido a la marca y el modelo del dispositivo, este valor sirve para realizar un análisis demográfico de los usuarios y ubicarlos dentro de grupos que comparten hábitos de uso.

#### 3.1.8. Attributed

Cada instalación tiene información de si esa instalación fue atribuida a *Jampp* o no, en el caso de los datos que teníamos, el análisis previo mostró que ninguna instalación fue atribuida a *Jampp*, pero igualmente lo agregamos como feature, porque en el caso de que halla 1 elemento si atribuido, creemos que sumaria mucho al objetivo de la competencia. Por ejemplo en otro set de datos.

### 3.2. Auctions

Para las subastas, el feature engineering se planteo con el objetivo de desarrollar la información provista por el csv de auctions en features que sean impactantes para la capacidad del algoritmo de machine learning en realizar las predicciones.

#### 3.2.1. Cantidad de Subastas

Cada entrada del archivo de auctions tiene información de una subasta y el usuario que vio esa subasta. Usando esto calculamos la cantidad de subastas que vio un usuario.

#### 3.2.2. Cantidad de Sources distintas

Se realizo un feature que da información de la cantidad de sources distintas que tienen las subastas de los espacios de publicidad que el usuario vio. Para encontrar esta feature se utilizo el comando nunique() de pandas que cuenta la cantidad de elementos únicos en una serie.

#### 3.2.3. Cantidad de Reference Types distintos

Los reference types daban a entender que podía haber mas de un usuario atribuido a un reference hash, para tomar esto en cuenta en la predicción, se realizo un feature que muestra la cantidad de reference types atribuidos a un mismo reference hash, se utilizo el comando nunique() de pandas para encontrarlo.

#### 3.2.4. Source mas representativa

Ademas de calcualr la cantidad de sources distintas que vio un usuario, se calculo la source ams representativa para un usuario, se utilizo la medida de la moda para encontrar ese valor.

### 3.3. Clicks

Para el csv de Clicks se realizo feature engineering para buscar features que complementen aquellos encontrados para Installs y Auctions.

#### 3.3.1. Cantidad de Clicks

Cada entrada del csv del archivo Clicks tiene información de un click realizado por un usuario, con esto se contó la cantidad de clicks que realizo un usuario.

#### 3.3.2. Clicks con Wifi

El archivo mostraba si el click realizado se había realizado con o sin wifi, sabiendo la fracción de clicks que se realizan con wifi podemos saber algo de lo hábitos del usuario, para esto se contó la cantidad de clicks que se realizaron con wifi.

#### 3.3.3. Promedio de tiempo para realizar un Click

Cada click viene con información del tiempo que tardo el usuario en hacer click una vez que se le muestra una publicidad, usando este tiempo se busco el tiempo promedio que tarda cada usuario en hacer click. De vuelta esto nos puede dar información de los hábitos del usuario ante las publicidades.

#### 3.3.4. Cantidad de Sources distintas

Para cada click se tiene información de cual fue la source del espacio de publicidad, se calculo a partir de esto la cantidad de sources distintas que vio un usuario.

#### 3.3.5. Cantidad de Advertisers distintos

Parecido a las sources, para cada click se tiene información de cual fue el advertiser que puso la publicidad, se calculo la cantidad de advertisers distintos que vio un usuario.

#### 3.3.6. Mean Location

Cada click viene con información de cual fue la altitud y la longitud en la que se realizo dicho click, combinando este valor y si es que el usuario tenia wifi, se puede saber algo de los hábitos del usuario en cuestión. Con los valores de latitud y longitud se calculo la posición media para cada usuario.

#### 3.3.7. Limitaciones de los clicks

Después de realizar el feature engineering para el archivo de clicks, y juntar la información con los installs, nos paso que ninguno de los reference hash coincidía, lo que limito el aporte que tuvieron los clicks al resultado de las predicciones.

### 3.4. Encoding de variables catégoricas

Como es sabido algunos algoritmos como XGBoost no aceptan features categóricos (unicamente ints, floats o booleans). Encontramos en los foros de Kaggle un algoritmo que nos dió buenos resultados al cual el autor lo llama TargetEncode pero que es en realidad una variante de Mean Encoding. Como vimos en clase, este método filtra los labels al set de entrenamiento, para atenuar el overfitting que esto produce el autor de TargetEncode utiliza Smoothing y además le agrega ruido. El link al algoritmo se encuentra en la sección de *Más Información.*

Utilizar dicho algoritmo mejoro muchísimo el RMSE, el hecho de permitirnos utilizar todo tipo de features mejoro completamente al modelo.

## 4. Entrenamiento y Predicciones

En esta sección describiremos los algoritmos de ML que hemos probado, como los hemos entrenado y cuales fueron sus respectivos scores. Son algoritmos en base a árboles de decisión ya que como vimos en clase generan buenos resultados en este tipo de problemas y además son eficientes. Hemos probado los siguientes algoritmos: XGBoost, Random Forest y Adaboost con Random Forest.

### 4.1. Primera Prueba

Para empezar utilizamos para entrenar todos los datos de los usuarios que se encontraban en el archivo installs.csv, pertenecieran o no al archivo de targets (disponible en la competencia en Kaggle). También utilizamos la ventana de tiempo del día 18 al 20 como primer set de entrenamiento y la ventana de tiempo del día 21 al 23 como primer set de test. Como segundo set de training utilizamos la ventana del 19 al 21 y como segundo set de test del 22 al 24 (seguimos asi hasta conseguir 4 pares de sets de train y test). Luego predijimos las variables time_to_install y time_to_auction para cada par de sets y como resultado final calculamos el

promedio entre los 4. Con nuestros datos el RMSE era mayor a 100.000, al igual que en Kaggle.

### 4.2. Segunda Prueba

Respecto a la primera entrega cambiamos los datos que utilizamos para entrenar, eliminamos los datos de los usuarios que no se encontraban en el archivo de targets. También decidimos utilizar únicamente 1 par de sets: 18 al 20 y 21 al 22 para entrenar y para testear respectivamente. En este caso, todos los modelos excepto XGBoost overfittearon, dando valores alrededor de 45.000 RMSE con nuestros datos y de mas de 100.000 RMSE en Kaggle. XGBoost logro lo que consideramos un primer buen resultado: 84.700 RMSE en Kaggle (mismo resultado con nuestros datos).

### 4.3. Tercera Prueba

Respecto a la segunda prueba, decidimos volver a utilizar los 4 pares de sets utilizados en la primera prueba. Este cambio produjo una significativa baja de RMSE en Kaggle, nuestro segundo buen resultado: 79.762.

### 4.4. Prueba Final - Mejor Score en Kaggle

Con respecto a la tercera prueba, agregamos como feature en cada paso de entrenamiento y predicción, las predicciones de las ventanas anteriores. Este agregado a la tercera prueba mejoro nuestro score en Kaggle bajando el RMSE a 79.579, nuestro mejor score en la competencia.

Se puede volver a realizar el submit descargando el código de nuestro repositorio en github, dentro de las carpetas tp2/preds_como_feature primero se corre el notebook FeatureEngineeringInstalls, luego MeanEncoderInstalls, luego ambos XGBoost (installs y auctions). Finalmente se corre el notebook llamado JoinPredictions que genera el archivo .csv final con las predicciones para cada usuario del archivo target.

NOTA: para los datos de auctions ya se encuentran realizados los primeros 2 pasos correspondientes a FeatureEngineering y MeanEncoding ya que realizarlos requiere utilizar una computadora potente.

## 5. TensorFlow

Luego de investigar distintos algoritmos a través de las diferentes web y comunidades de desarrolladores como por ejemplo stackoverflow.com, se llegó a la web del algoritmo TensorFlow (https://www.tensorflow.org/). En dicha web pudimos encontrar material suficiente y de sobra para entender el concepto del algoritmo y tutoriales para aplicar el algoritmo en set de datos pequeños. Luego de leer las diferentes guias y realizar los diferentes tutoriales se implementó el algoritmo en nuestro set de datos. La implementación del algoritmo se hizo de forma similar tanto para predecir las conversiones como las próximas muestras.

### 5.1. Datos

Tanto para los installs, los clicks y las auctions se dividieron los set de datos en 7 grupos diferentes dependiendo de la fecha: Del 18/04/2019 al 20/04/2019; Del 19/04/2019 al 21/04/2019; Del 20/04/2019 al 22/04/2019; Del 21/04/2019 al 23/04/2019; Del 22/04/2019 al 24/04/2019; Del 23/04/2019 al 25/04/2019; y del 24/04/2019 al 26/04/2019.

En los diferentes archivos nos quedamos solo con la información que consideramos relevante o creamos otros que consideramos que iban a ser de utilidad para el algoritmo.

Para el archivo de installs nos quedamos con los siguientes features: [ref_hash, cant_installs, device_brand, device_model, application_id, cant_apps, user_agent, device_language, cant_clicks, cant_wifi_clicks, mean_time_to_click, cant_sources_clicks, cant_advs_clicks, mean_location, time_to_install.]

Para los archivos de auctions: [ref_hash cant_auctions cant_sources cant_types time_to_auction].

Para los archivos de clicks: [ref_hash cant_clicks cant_wifi_clicks mean_time_to_click cant_sources_clicks cant_advs_clicks mean_location]

La idea de generar estas ventanas es poder entrenar el algoritmo con una de las ventanas y testearlo con la ventana siguiente. Por ejemplo: entrenamos el algoritmo de ML para predecir los installs con el archivo de installs del 18/04/2019 al 20/04/2019 y luego testeamos en algoritmo con el archivo de install del 19/04/2019 al /20/04/2019.

Con este método logramos tener 6 distintas versiones del mismo algoritmo entrenado con diferentes set de datos. Luego de obtener la prediccióǹo de los 6 algoritmos, se promedio el resultado de todos los algoritmos para obtener el valor final de la estimación.

### 5.2. El Algoritmo

Una vez ya procesados los diferentes archivos y generado las diferentes ventanas se llevaron a cabo las predicciones tanto del tiempo hasta la próxima aparición como del tiempo de conversión de cada usuario.

Lo primero que se realizó fue la lectura de los diferentes archivos mediante pandas.

Una vez que tenemos todos los archivos de las diferentes ventanas cargados se lleva a cabo la limpieza y depuración de lo mismo, borrando columnas innecesarias y completando los NaN con valores posibles.

Se definieron los diferentes archivos de train y test para las 6 diferentes pares de ventanas. Se separaron estos archivos de train y test en features y label (valor conocido a estimar) y luego se procedió a realizar la normalización de los datos ya que el algoritmo funciona bastante mejor si los datos se encuentran normalizados.

Se construyeron 6 modelos secuenciales utilizando 3 dense layers, los dos primeros de 64 celulas mientras que el último tan solo de 1 celda (Ya que hacemos una regresión y queremos un solo valor de salida).

Se probaron diferentes layer, tipo de modelo y número de células pero los modelos se hacían muy costosos, caían en overfitting más fácilmente y no era conveniente cambiar del modelo sequencial, con layers de densidad de 64, 64 y 1 células.

Una ves definidos los 6 diferentes modelos, se procedió a entrenar el algoritmo con los diferentes archivos de train y se testearon los resultados con el archivo de test como así también se llevó a cabo la predicción de los valores del archivo de test.

Una vez que obtuvimos los 6 valores diferentes para una misma predicción, se realizó un promedio de todas y se guardó en un archivo de predicciones, ya sea para installs como para

auctions ya que el algoritmo es muy similar para ambas.

## 6. Conclusiones

Luego de probar distintas variaciones de algoritmos de ML y sets de entrenamiento y test podemos afirmar que para lograr un buen resultado a la hora de querer predecir el valor de una variable se deben tener en cuenta 3 partes del proceso de ML por sobre el resto.

La primera es el encoding de variables categóricas, es importante ya que disminuyó el RMSE aún teniendo features simple. Por recomendación de los profes, decidimos utilizar mean encoding por sobre one hot encoding ya que este ultimo genera una columna por cada valor posible de la variable, lo cual puede generar sets de datos muy pesados.

La segunda son los features. Por lo que explicamos anteriormente, el hecho de poder utilizar features categóricos mejoro mucho al modelo. También se explica la diferencia entre los RMSE de predecir auctions e installs con los datos locales por el hecho de haber encontrado mejores features para auctions que para installs.

Finalmente la elección del algoritmo de ML a utilizar. En nuestro caso nos quedamos con XGBoost debido a su resultado, que es lo más importante para una competencia pero además este algoritmo es muy rápido a la hora del entrenamiento por lo que para nosotros en este caso la elección era clara.

## 7. Más Información

Nuestro análisis fue realizado en Python Pandas. Utilizamos un repositorio público de github para juntar análisis, feature engineering y entrenamientos de ML. El link del repositorio es:

https://github.com/GastonMontes/75.06-Datos-Grupo22-TP1

Dentro del repositorio todo lo trabajado para el TP 2 se encuentra en la carpeta tp2. Los datos no fueron incluidos en el repositorio por ser muy pesados, se pueden encontrar en el siguiente link:

https://drive.google.com/open?id=1-HKn0Pw4irUVrK2rrYFsxkjcsG5C3YNa

El enunciado del trabajo práctico se puede encontrar en el siguiente link:

https://docs.google.com/document/d/1aJUqtO-9QlHLT7jCcRUQC5TD4JX7VipFe0C4pUO2Gbg/edit

El link a la competencia en Kaggle es el siguiente:

http://www.kaggle.com/c/jampp-at-fiuba-competition/

El link a TargetEncode (algoritmo de Mean Encoding):

https://www.kaggle.com/ogrellier/python-target-encoding-for-categorical-features

Link a la web de TensorFlow:

www.https://tensorflow.org