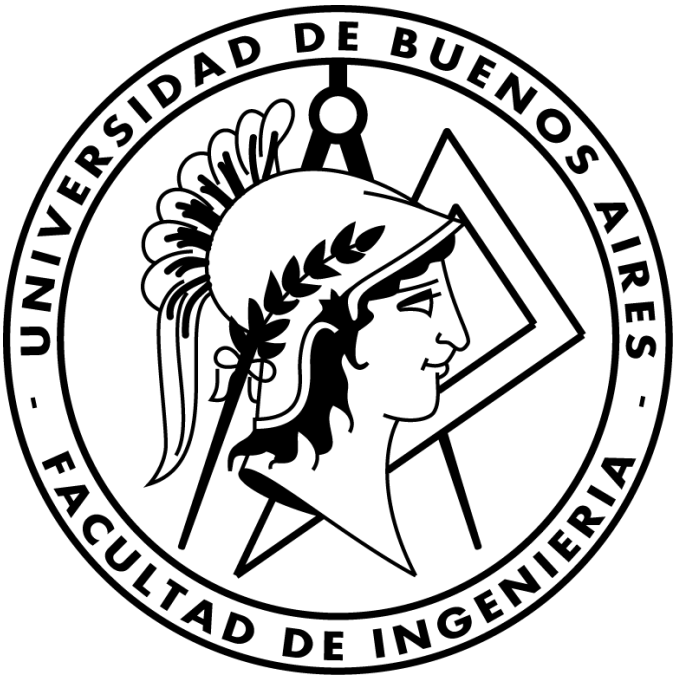
Trabajo Práctico 2

75.06 - Organización de Datos



Grupo 31

"El mejor equipo de los últimos 50 datos"

Alumnos:

Nombre	Padrón
Gatti, Nicolás	93570
Del Carril, Manuel	100772
Verón, Lucas	89341

Github link: https://github.com/gatti2602/7506-datos/tree/master/tp2

Índice

Introducción

El presente trabajo tiene como objetivo la predicción de los tiempos hasta la próxima aparición en un dispositivo (St) y también de su conversión (St), en ambos casos este tiempo se medirá en segundos desde el comienzo del día siguiente al último día de la muestra de datos.

El trabajo se organizó de la siguiente manera:

- Filtrado y arreglo de datos: Dado el tamaño de los datasets fué necesaria la conversión de los tipos de datos campo por campo optimizando el tamaño en memoria de cada variable. Se revisaron además los campos que no eran necesarios para no importarlos. Esto fue vital ya que las operaciones realizadas sobre los mismos exigian al máximo los equipos de desarrollo disponibles.

  Para el filtrado de los datos se tuvieron en cuenta algunas inconsistencias en los datos observados.

- Armado de Features: Cada feature puede entenderse como un dato agregado asociado a los dispositivos que se desean convertir. Para el armado de los mismos se tuvieron que tomar decisiones claves que afectaron el desarrollo posterior de todo el proyecto. Por ejemplo, tamaño de las ventanas de muestra de datos y como completar o transformar los valores obtenidos para cada dispositivo.

- Armado de Labels: Dado que no se proveyó de un conjunto de labels estable, como parte del proyecto se incluyó el armado de los mismos que se realizó, obviamente en sintonía con las ventanas de features armada. Dado que esta definición fue realizada por el equipo la misma afecta a las predicciones realizadas.

- Entrenamiento del modelo predictivo: Utilizando los features y los labels se probaron diferentes modelos predictivos. Dado que el objetivo era un modelo de regresión, se utilizaron principalmente modelos de Árboles. El tuning del mismo se realizó con Grid Search durante las pruebas.

A continuación desarrollaremos la implementación elegida para la resolución de este problema además de todos los impedimentos y traspiés ocurridos. También describiremos el proceso de *feature engineering* que llevamos a cabo.

Diseño e implementación

Organización del set de datos

Para poder llevar a la generación del modelo predictivo es necesario contar con una buena cantidad de features("columnas") que permitan entrenar el modelo correctamente.

Previo a la generación de features es necesario llevar a cabo la transformación de los "datos en crudo" que provienen de los archivos csv suministrados. Dado el tamaño de los mismos, elegimos los tipos acorde para un efectivo uso de memoria.

Aún cuando definimos tipos, era fácil caer en un Memory Error, ya que las demandas de memoria superaban la de los equipos, o la corrida de Notebooks tardaba mucho tiempo. Por eso, una buena parte del tiempo estuvimos trabajando con fracciones aleatorias del csv original. Solamente comenzamos a utilizar todo el set una vez que tuvimos que entrenar el modelo donde era necesario el uso completo de los datos.

En primer lugar, decidimos organizar todos los dispositivos que aparecen en los distintos csv en un csv único, el cual usamos como identificador de los mismos.

Aquellos dispositivos que se veían duplicados, es decir que el mismo identificador aparecía para distintos tipos de dispositivo fueron eliminados, ya que la información no era confiable.

Durante el desarrollo, dividimos el set de datos en ventanas de 3 días, incentivados por las recomendaciones del enunciado. Del 18 al 20, del 21 al 24 y 24 a 26. Estos intervalos de tiempo son independientes unos de otros, es decir, no hay solapamiento entre los mismos.

Todos los features tomados fueron calculados para de forma independiente para cada ventana (salvo algunas excepciones que por definición no eran posibles).

Aprovechando estas ventanas fue sencillo armar el dataset de labels, pués bastó mirar la primer aparición del dispositivo en la ventana siguiente para determinar los tiempos St y Sc.

Modelo predictivo desarrollado

Para construir los modelos se ensayaron distintos modelos predictores, donde los predominantes fueron modelos con árboles siendo XGBoost el que obtuvo mejores resultados coincidiendo con lo esperado.

Modelos ensayados:

- Random Forest: Para primeras pruebas y determinación de features importantes
- XGBoost: Aplicado como Random Forest pero con mejores resultados. Se utilizó Grid Search para optimización de hiperparametros, sin embargo no se obtuvieron mejoras considerables frente a los valores por defecto del algoritmo.

- KNN: Se utilizó como modelo alternativo a la predicción con Árboles. Sin embargo, no se obtuvieron niveles de precisión comparables. Se observó que el modelo se ajustó mucho mejor a los valores de prueba obteniendo errores menores y más parejos, sin embargo en la predicción final tuvo resultados peores, por lo que se estima que el modelo pudo haber overfiteado.

El modelo predictor final es un ensamble de distintos resultados:

Con las primeras dos ventanas, de las que se disponen labels, se entrena un modelo distintos para tiempos de arribo y otro para installs, entendiendo que las decisiones para llegar a cada resultado deben ser independientes.

Luego con los datos de la tercer ventana se realizan las predicciones finales. De esta forma no hay que normalizar los datos por el uso de ventanas de tiempo desiguales.

Una vez se realizan las predicciones requeridas, se completaron los equipos faltantes con los tiempos promedio de las ventanas anteriores. Esto significa que en los casos donde en los últimos 3 dias el dispositivo no apareció se toma como predicción el valor conocido de las ventanas anteriores. La otra alternativa ensayada fué predecir el máximo tiempo posible para estos equipos, sin embargo no se lograron buenos resultados y solo se utilizó para predecir los dispositivos que no aparecen en ningún dataset (18 en total).

El modelo final entrenado en cada ventana es un XGBoost. Para sus parámetros realizamos un pequeño GridSearch.

Previo a la idea de utilizar un modelo por ventana, la idea era ir entrenando el modelo con cada ventana e intentar predecir lo que sucedía en la siguiente. De esta manera, llegábamos a la tercera ventana la cual utilizamos para predecir los valores pedidos en la competencia. Dado que el score obtenido no era bueno, nos pusimos a investigar cuál podría ser el problema y descubrimos que los *targets* eran poco frecuentes en la ventana 3.

Por esta razón, decidimos entrenar un por modelo por ventana, para aprovechar todos los datos a la hora de predecir los *targets*. Esto mejoró nuestro score en 4 mil puntos.

Como comentario final, se realizaron pruebascon distintos tipos de combinaciones de datos que resultaron en scores mas bajos:

- Un solo juego de datos compuesto con features de los primeros 6 días y los ultimos 3 para los labels.
- Entrenar con ventanas de 3 dias pero predecir con los datos completos.

Feature Engineering

Una vez propuesto el algoritmo de Machine Learning a utilizar, nos enfocamos en la producción de features. Cada uno a medida que se le ocurría un posible feature lo programaba y lo compartía con el resto del equipo. Nos dispusimos a probar el rendimiento de los mismos una vez tuviésemos una cantidad interesante. Los features diseñados se listan a continuación con una breve descripción:

- ***Cantidad de Sources: amount_dif_src (****Auctions* ***dataframe)***
  El feature es la cantidad de fuentes(sources) diferentes que de los que proviene cada dispositivo. La idea de este feature es que mientras más sources tengan al dispositivo en sus listas más probabilidades hay que se le ofrezca una subasta.

- ***Cantidad de Encuestas Pasadas: auctions_total (****Auctions* ***dataframe)***
  Cantidad de apariciones de cada dispositivo en total dentro de todo el dataframe de Auctions. Lógicamente se esperaba que mientras más veces aparezca el dispositivo más probable sea que vuelva a aparecer.

- ***Cantidad de encuestas en la última hora: auctions_last_hour (****Auctions* ***dataframe)***
  Cantidad de apariciones en la última hora. Este feature tiene hipótesis implícita de que si un dispositivo aparece con frecuencia es más probable que vuelva a aparecer en poco tiempo.

- ***Tiempo entre encuestas medio: secs_to_next_mean (****Auctions* ***dataframe)***
  Tiempo medio entre apariciones en subastas. Simplemente el tiempo medio entre apariciones de los dispositivos, si consideramos que los tiempos de arribo siguen un proceso de Poisson , el tiempo medio es el unico parametro necesario para describir los tiempos de arribo.

  Esta métrica se agrego para todos los datasets y se incluyeron adicionalmente los **Máximo, Mínimo y Desviación.**

- ***secs_since_last_arrival (****Auctions* ***dataframe)***
  Tiempo transcurrido desde la última aparición en una subasta. Es ir hacia atrás respecto de lo que queremos predecir. Usando el tiempo de las apariciones pasadas para predecir las apariciones futuras.

- ***amount_auctions_in_weekend (****Auctions* ***dataframe)***
  Cuántas veces aparece cada dispositivo en una subasta los fin de

semana(días 20 y 21).
Al no proveer una mejora y dado que solo una ventana incluye el fin de semana, no se incluyó en los resultados finales.

- ***Última aparición fue en fin de semana: is_last_weekend (****Auctions* ***dataframe)***
  Indica si el último día de la ventana de tiempo(ventana de 3 días) se corresponde a un día de fin de semana o no.

- ***Cantidad de veces que aparece en horarios de alta demanda: alta_demanda (****Auctions* ***dataframe)***
  Como del análisis exploratorio se pudo ver que las apariciones tienen "rangos" de horarios en donde es hay más apariciones, se indica si el dispositivo apareció dentro del horario de "alta_demanda" o de muchas apariciones que es: [0,2] hs.

- ***Agrupado de horas: pond_hora_ext (****Auctions* ***dataframe)***
  Teniendo en cuenta los rangos de horarios de apariciones por cantidad se pondera de acuerdo a que rango pertenece el dispositivo y su aparición. Rangos: [1,2] = 0.90 ; [20,0] u 3 = 0.70 ; [11,20] u 4 = 0.5 ;[5,10] = 0.20.

- ***Cantidad de Eventos: amount_events (****Events* ***dataframe)***
  Cantidad de cada evento para el dispositivo en cada ventana.

- ***Accesos con Wifi: wifi (****Events* ***dataframe)***
  Indica si el dispositivo tiene uso o no wifi.

- ***Ponderación por día: pond_M_X_ev (****Events* ***dataframe)***
  Ponderación por día Martes o Miércoles. De acuerdo al día de la semana se indica que el día fue Martes ó Miércoles(True) u otro día(False). Esto último resulta del análisis exploratorio donde los días Martes y Miércoles había más eventos.

- ***Tiempo hasta último, anteúltimo y ante anteúltimo evento: last_event, before_last_event, before_before_last_event (****Events* ***dataframe)***
  Indica los tiempos en segundos transcurridos entre respecto de los últimos 3 eventos a partir del evento del registro.

- ***Cantidad de app distintas instaladas: count_app_dif (****Installs* ***dataframe)***
  Cantidad de application distintas instaladas por el dispositivo.

- ***Instaló en la ventana anterior: win_2_in_1,win_3_in_2 (****Installs* ***dataframe)***
  Cada columna(win_2_in_1, win_3_in_2) indica si el dispositivo que pertenece a una ventana instaló en una ventana anterior(2 en 1, 3 en 2).

- ***Cantidad de tipos por 'ref_hash': count_for_kind (****Installs* ***dataframe)***
  Cantidad de tipos diferentes, reemplazando el kind(category) del cada dispositivo.

- ***Cantidad distintos de Advertisers: amount_dif_advertisers*** *(Clicks* ***dataframe)***
  Cantidad de anunciantes distintos que contiene el dispositivo.

- ***Tiempo promedio de clicks: timeToClick_mean*** *(Clicks* ***dataframe)***
  Tiempo promedio de clicks de cada dispositivo. No resultó de utilidad para las predicciones

- ***Armado de Clusters: KMeans20, KMeans50***
  Clustering de todos los features obtenidos usando KMeans se armaron columnas con distinta cantidad de clusters desde 10 a 100, no se obtuvo mejora en las mediciones utilizando los mismos. Se intentó obtener lo mismo por Spectral Clustering no pudiendo procesar el dataset de features en memoria.

  Todos estos features fueron calculados para el set de entrenamiento y todos resultaron útiles.

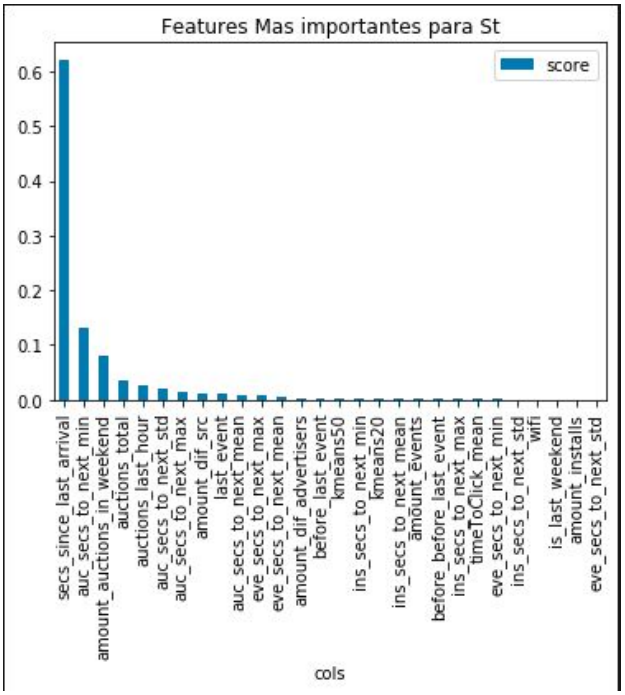
  Los siguientes features fueron probados, pero algunos no generaron una mejora significativa, motivo por el cual, se decidió quitarlos del modelo final. Alguno de los features son los siguientes:

  alta_demanda, count_app_dif, win_2_in_1, win_3_in_2, count_for_kind

Importancia de los Features

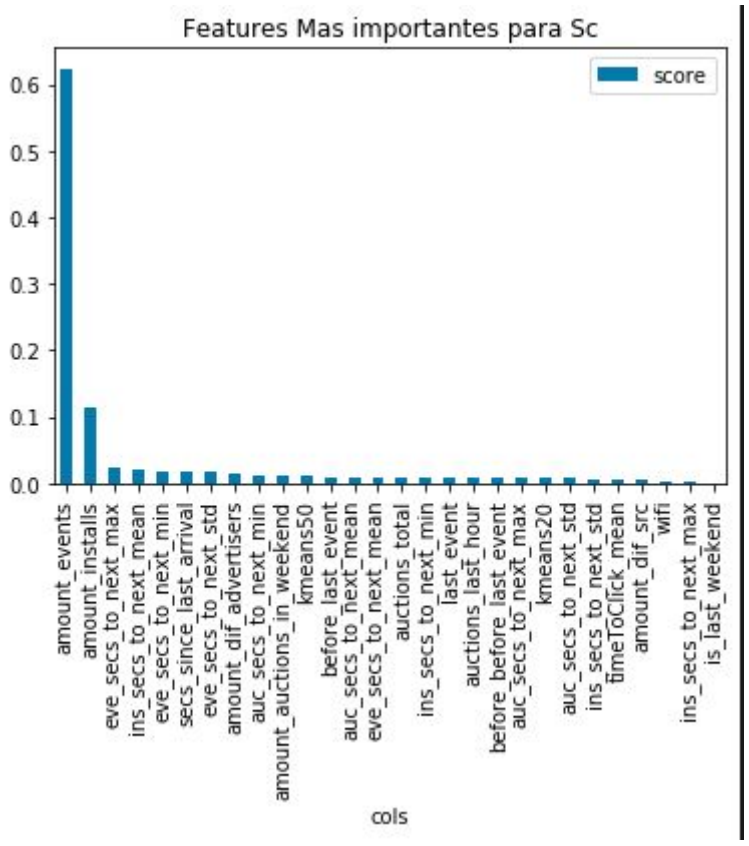
Se adjunta importancia de todos los features segun KGBoost y según el target deseado:

- Tiempos hasta nueva aparición:



Se puede ver que para la predicción de los tiempos de arribo todos los features de mayor importancia están relacionados con la cantidad de apariciones en el dataframe de encuestas.

- Tiempo hasta Install:



En este caso se visualiza que los features más útiles son la cantidad de eventos y de installs de los dispositivos. Esto no es muy descriptivo, por lo que se entiende que los features obtenidos no terminan de ser útiles para predecir los installs.

Comentarios Finales y Conclusiones

Inicialmente se comenzó tratando de resolver el problema de las "ventanas" de tiempo en los cuales se producen los eventos(apariciones, instalaciones) para poder entrenar el modelo. Dada la separación de las ventanas en intervalos de 3 días, se usó la ventana 1 para predecir los targets(apariciones,instalaciones) de la ventana 2; luego la ventana 2 y se predijo en la ventana 3 y finalmente se usó la ventana 3 para predecir los targets finales.

El primer submit permitió ingresar a la competencia con un score aproximado de 166 mil.

Se aumentaron la cantidad de features en el modelo de regresión(Random Forest) y se obtuvieron mejoras en el score final, aunque si se obtenían mejoras cuando se entrenaba el modelo. Se debía aumentar la cantidad de features(inicialmente 6) ya que los mismos no permiten bajar el puntaje.

Se agregaron más features para mejorar el modelo y adicionalmente se utilizó un modelo de **Xgboost** para hacer la regresión. Esto último permitió bajar el score a 161k.

Un problema recurrente y que llevó a muchas demoras fue el tamaño de los dataframe. Hacer frente a esa cantidad de datos muchas veces llevo a que las máquinas llevaran varios minutos procesando un dataframe ó un transformación en particular.

Para limitar este problema durante las pruebas iniciales se muestreó aproximadamente un 40% de los dataframes y se armaban y probaban los modelos. Sin embargo, esta tarea ocultó varios problemas en los datos que impedían llevar el modelo desarrollado al tamaño real.

Se realizaron varias iteraciones y optimizaciones en el proceso de extracción de features y arreglo de datos de manera de minimizar el espacio en memoria:

- Supresión de campos no utilizados.
- Optimización de tamaño de variables.
- Categorización de Objects.
- Supresión de datos erroneos.
- Optimización de las funciones de Pandas utilizadas, sobre todos en GroupBy.

Durante las pruebas no se utilizaron modelos de redes neuronales dado que se estimó que los features obtenidos no podían interoperarse, sino que formaban parte de un set que permite clasificar los dispositivos en grupos de acuerdo a su comportamiento y hacer la regresión de esta forma. Por esta razón se optó por modelos de árboles y KNN, siendo los primeros los que arrojaron los mejores resultados.

La decisión de separar el dataset en ventanas iguales, tiene sentido para un modelo de producción, sin embargo durante la competencia resultó que muchos dispositivos no tenían datos en los últimos tres días. Para minimizar esto se utilizó datos de las ventanas

anteriores para predecir los. Esto utiliza el conocimiento de lo que sucedió en el pasado pero no aplica ningún modelo predictivo para la predicción.

Finalmente se obtuvieron mejores resultados utilizando el promedio de todos los features obtenidos para los dispositivos en un solo modelo predictivo.

Instructivo de Ejecución

Una vez descargado el repositorio correr en orden los notebooks en la carpeta Final. En orden, producirán los siguientes resultados:

1. Crea el listado de refhash únicos válidos, descartando aquellos que están duplicados.
2. Arma los labels de cada ventana a predecir. Genera el archivo *targets.csv*.
3. Arma los features: Tarda aproximadamente media hora en correr y genera el archivo training_set.csv como salida.
4. Predicciones, cada archivo ejecuta un modelo predictivo con diferentes estrategias: Diferentes modelos o diferente forma de agregar los features.