Facultad de Ingeniería U.B.A.

75.06– 95.06 Organización de Datos

“Predicción de conversión de un usuario de trocafone”

TP 2 - Informe

Segundo Cuatrimestre 2018

Apellido y nombre	Padrón
Alfonso Amezqueta	94732
Mejia Cordova,Yussef Omar	92.963
Villani, Cristian Daniel	93.358

1) Preparación del set de entrenamiento

Features temporales:

- **Feature frecuencia de eventos por cada día de la semana:** creamos columnas para cada dia de la semana y se rellenan con la cantidad de eventos que realizo un usuario por cada día de la semana, sería una distribución de los eventos en los 7 días de la semana. Así se puede ver como se maneja el usuario diariamente. En los data frames finales estos features reciben el nombre :
    - **Lunes**
    - **Martes**
    - **Miercoles**
    - **Jueves**
    - **Viernes**
    - **Sabado**
    - **Domingo**
- **Feature frecuencia de eventos por cada mes:** creamos columnas como las explicadas en el ítem anterior para los meses. Así se puede ver como se maneja el usuario mensualmente. Sus nombres son:
    - **Enero**
    - **Febrero**
    - **Marzo**
    - **Abril**
    - **Mayo**
    -
- **Feature tiempo desde el último evento hasta fecha límite:** para cada uno de los eventos se busco la distancia entre el momento en que realizó ese evento específico y la fecha límite que es el 31 de mayo ese valor en segundos se agregó a once nuevas columnas, si el usuario no realizó dicho evento se le asigna un -1 como forma de distinguir. Este resulto ser un feature muy poderoso y nuestro modelos empezaron a rankear más alto luego de incorporarlo. Los eventos que recibieron más importancia fueron checkout, viewed product y visited site, esto tiene sentido ya que el usuario que realiza vistas o checkout en un momento muy próximo a la fecha límite denota una intención de compra y tiene más probabilidades de realizar una conversión en la misma. En cambio eventos como 'lead' fueron rankeados muy abajo en la importancia. Sus nombres y los eventos a los que pertenecen son:
    - **time_to_last_vs (visited site)**
    - **time_to_last_vp (viewed product)**
    - **time_to_last_ch (checkout)**

- **time_to_last_co (conversion)**
- **time_to_last_le (lead)**
- **time_to_last_ad (ad campaign hit)**
- **time_to_last_st (staticpage)**
- **time_to_last_bl (brand listing)**
- **time_to_last_gl (generic listing)**
- **time_to_last_se (search engine hit)**
- **time_to_last_sp (searched product)**

- **Feature tiempo desde el primer evento hasta fecha límite:** idéntico al feature que explicamos antes pero ahora para la diferencia de tiempo límite con el momento donde el usuario realizó el primer evento de cada tipo. Como su contraparte se le asignó -1 a aquellos usuarios que no realizaron el evento, pero a diferencia de la misma la importancia no fue tan alta situándose más abajo incluso para eventos como checkout. Los nombres que reciben en el dataframe son:
  - **time_to_first_vs (visited site)**
  - **time_to_first_vp (viewed product)**
  - **time_to_first_ch (checkout)**
  - **time_to_first_co (conversion)**
  - **time_to_first_le (lead)**
  - **time_to_first_ad (ad campaign hit)**
  - **time_to_first_st (staticpage)**
  - **time_to_first_bl (brand listing)**
  - **time_to_first_gl (generic listing)**
  - **time_to_first_se (search engine hit)**
  - **time_to_first_sp (searched product)**

Features categorizable:

- **Feature de eventos basado en frecuencia**: un encoding basado en frecuencia para la variable categórica evento, consistió en agregar 11 columnas para cada evento y en cada una de ellas se pone para cada usuario cuantas veces se realizo dicho evento. Se los puede identificar porque llevan el nombre de cada evento al que pertenecen, o sea, conversion, checkout, viewed product, etc, y asi para cada evento.
- **Feature de marcas basado en frecuencia:** consistió en para cada usuario obtuvimos las marcas de los modelos para cada celular para todos los eventos que realizaron que poseen el feature model, de esta manera para cada usuario agregamos una columna por cada marca y contabilizamos la frecuencia de aparición por usuario de las mismas. En en dataframe tienen el nombre de:
  - **Samsung**
  - **iPhone**
  - **Motorola**
  - **Sony**

    - **Lenovo**
    - **LG**
    - **Asus**
    - **iPad**
    - **Novo**

- **Dos features de storage y condition:** realizamos features de encoding basados en frecuencia de la misma manera descrita en los dos ítems anteriores para, almacenamiento de los dispositivos y el campo condition. Para los relacionados con almacenamiento, sus nombres son:
    - **4GB**
    - **512MB**
    - **256GB**
    - **8GB**
    - **32GB**
    - **64GB**
    - **128GB**
    - **16GB**

  Para los relacionados con condition, los nombre son:
    - **Muito Bom**
    - **Bom**
    - **Bom - Sem Touch ID**
    - **Excelente**

- **Mean_encoding sobre los campos city, region, country, device type, browser system, operating system version y search engine:** como su nombre lo indica son 7 mean encoding realizados sobre los campos mencionados, el campo label fue utilizado como **label** del mean encoding. No se realizó smothing o algún método de control de regularización para ellos por lo que para algunos modelos tendieron mucho al overfitting, en especial si se usaba el encoding relacionado con el campo city. Aun así en XGBoost y sin el campo city y con gridsearchCV para tunear los hiperparametros pudimos lograr un summit con 0.85 en kaggle. Estos features no se encuentran en el .csv con el que logramos la mejor puntuación pero agregamos un csv donde si estan adjuntos. Los nombres son:
    - **city_mean**
    - **region_mean**
    - **engine_mean**
    - **country_mean**
    - **system_mean**
    - **browser_mean**
    - **device_mean**

Feature estadísticos:

- **Feature evento más popular:** indica el nombre del evento que más realizo cada usuario. Su nombre en el dataframe es "*most_frequent_event.*" Este objeto será encodeado binariamente y cuatro nuevas columnas surgirán del mismo, las mismas son "*most_frequent_event I, II, III y IV.*".
- **Feature cantidad de veces que el usuario realizó el evento más popular:** para ese evento en particular decimos cuántas veces fue realizado. En el dataframe tiene el nombre ***"count_most_frequent_event".***
- **Feature evento menos popular:** indica el nombre del evento que menos realizó cada usuario. Al igual que su contraparte este feature fue encodeado binariamente en cuatro nuevas columnas. En el dataframe se identifica con el nombre "*less_frequent_event.*", Las mismas son "less*_frequent_event I, II, III y IV.*"
- **Feature cantidad de veces que el usuario realizó el evento menos popular:** para ese evento en particular decimos cuántas veces fue realizado, su nombre en el dataframe es "*count_less_frequent_event*".
- **Feature del promedio de cantidad de eventos por usuario:** a las cantidades totales de cada evento que realizó cada usuario se le realiza un promedio. En el dataframe recibe el nombre de "*mean_per_event".*
- **Feature tiempo promedio entre cada evento para un usuario:** en este feature vemos cual es la distancia de tiempo para cada evento que realizó el usuario y luego los promediamos. Sirve para indicar con qué regularidad promedio el cliente realiza eventos. Su nombre es "*mean_time_between_events".*
- **Feature tiempo promedio entre cada tipo de evento para un usuario:** en este feature agregamos 11 nuevas columnas a nuestros set de entrenamiento y predicción, cada una de ellas por cada evento. En cada uno de ellos vamos a calcular la diferencia de tiempo promedio entre un mismo evento para cada usuario. Si el usuario realizó dicho evento una sola vez, como no hay un segundo evento con el cual comparar, a ese usuario se le asignará el valor 0. Si el usuario no realizó tal evento entonces se le asignará el valor -1 como forma de diferenciar. Para algunos modelos como el XGboost estos features alcanzaron una importancia alta, en especial para los eventos checkout, visited site y viewed product. Con estos datos podemos ver la distancia de tiempo promedio para un mismo evento por usuario. Para cada uno de los eventos, los nombres que reciben son:
    - **mean_time_between_vp (viewed product)**
    - **mean_time_between_ch (checkout)**
    - **mean_time_between_vs (visited site)**
    - **mean_time_between_co (conversion)**
    - **mean_time_between_st (staticpage)**
    - **mean_time_between_gl (generic listing)**

- **mean_time_between_bl (brand listing)**
- **mean_time_between_ad (ad campaign hit)**
- **mean_time_between_le (lead)**
- **mean_time_between_se (search engine hit)**
- **mean_time_between_sp (searched product)**

<u>Interacción entre features:</u>

- **Feature de multiplicacion:** multiplicacion de cantidad de evento checkout con de cantidad de evento convertion. En el dataframe se llama "multiplicacion_checkout_y_conversion".
- **feature de suma:** sumamos la columna cantidad de converciones con la columna del mismo tipo checkout. En el dataframe se llama "suma_checkout_y_conversion".
-

Features basados en texto (contains)

- usamos un wordcloud para visualizar las palabras más buscadas en search term. Colocamos como columnas,y pusimos las frecuencia de las palabras que buscaban en las mismas.

2) Modelos Machine Learning:

Una vez que preparamos los datos, nos toca elegir un modelo según nuestro objetivo,en nuestro caso necesitamos un modelo de aprendizaje supervisado regresivo, ya que queremos predecir la probabilidad de que los usuarios realicen una compra a futuro.

Modelos que usamos y una breve definición de mismos:

XGBoost:

Es un algoritmo desarrollado muy recientemente por Tian Chen, que es el estado del arte en los algoritmos de aprendizaje supervisado dando muy buenos resultados en las competencias.
Los hiperparametros que usamos del mismo son:

- **learning rate**: por default es 0.3, se encarga de hacer al modelo más robusto al encoger el peso en cada paso.
- **min_child_weight**: por default es 1. Define la suma mínima de pesos de todas las observaciones requeridas en un hijo. Los valores muy altos hacen que el modelo tiende al underfitting.
- **Max_deph**: es la máxima profundidad del árbol, por dafault es 6.
- **gamma**: hace al algoritmo conservativo.El nodo hace un split solo cuando este causa una reducción positiva en la función de pérdida.
- **subsample**: marca la fracción de observaciones a ser tomadas al azar en cada árbol.

- **colsample_bytree:** marca la fracción de columnas a ser tomadas al azar en cada árbol.
- **lambda**: el término de regularización L2 para los pesos.
- **alpha**: el término de regularización L1 para los pesos.
- **n_estimators**: número de árboles a usar.

K-Nearest Neighbours:

Se basa en encontrar para un determinado punto sus K-vecinos más cercanos y a partir de ellos decidir el output de lo que queremos predecir, por ejemplo utilizando el promedio del output de esos k más cercanos. Lo que hay que decidir en KNN es cuantos vecinos tomar y además como buscar esos vecinos, es decir, la función de distancia. El hiper parámetro que usamos es el número de vecinos k que el algoritmo debe tomar.

Red Neuronal

Las redes neuronales son utilizadas cuando se quieren reconocer patrones. Está compuesta por "neuronas" representadas por booleanos y cada booleano está asociado a una determinada cantidad de aristas que determinan el estado de cada neurona.

Las neuronas están organizadas en distintas capas, la de input, la de output, y muchas capas ocultas intermedias. Las neuronas de input están conectadas a las de la siguiente capa, y así hasta llegar a la capa de output. Las neuronas que están activadas afectan a los pesos que determinan si se activarán o no las neuronas de la capa siguiente, que afecta a los pesos de la siguiente capa, y así hasta llegar a la capa de output. Si la predicción es incorrecta, se vuelve atrás retocando los pesos de neuronas en distintas capas para llegar a un mejor resultado.

Creímos que era buena idea probar una red porque suele generar buenos resultados. Los hiperparametros son:

- activation: funcion de activacion para las capas, en nuestro caso ‘relu’.
- solver: es el solucionador por la optimización de peso;como se usa ‘lbgfs’, se usa un optimizador de la familia de métodos cuasi_newton.
- alpha: parámetro de penalización L2.
- hidden_layer_sizes: son los n elementos de la capa.
- random_state: es la seed usada por el número de de generadores random.

Random Forest:

Los random forest están compuestos por una cantidad determinada de árboles de decisión.

Para determinar el valor a predecir de un nuevo input, el mismo pasa por distintos arboles de decision que son armados cada uno utilizando distintos features, el resultado final es una ponderación de los resultados de los distintos árboles.
Es una modificación sustancial de bagging, la idea esencial del bagging es promediar muchos modelos ruidosos pero aproximadamente imparciales, y por tanto reducir la varianza. Los árboles son los candidatos ideales para el bagging, dado que ellos pueden registrar estructuras de interacción compleja en los datos, y si crecen suficientemente profundo, tienen relativamente baja parcialidad. Producto de que los árboles son notoriamente ruidosos, ellos se benefician grandemente al promediar.
Los hiperparametros son:

- max_depth: maxima profundidad de cada arbol.
- min_samples_leaf: el mínimo número de muestras necesarias para estar en un nodo hoja.
- n_estimators: número de árboles usados.

En todos los casos utilizamos grid search con cross validation para obtener los mejores hiperparametros, y también un feature selection, basado en forward selection, partimos del feature con mayor importancia entregado por el random forest importance features, el cual era checkout y en procedimos a entrenar el modelo y medir con la métrica AUC, en caso de que el nuevo feature probado mejorará el score lo dejabamos en caso contrario era descartado, al utilizar esta forward selection notamos que nos quitaba demasiados features, algunos de ellos al ser quitados nos reducían el score en kaggle, por ese motivo decidimos no usarlo.

XGBoost:
Fue el mejor en performance.
learning_rate=0.1,gamma=0,max_depth=3,min_child_weight=6,colsample_bytree=0.4 reg_alpha=1e-05, reg_lambda=0.45, subsample=0.95 , n_estimators = 100
score = 0.86582

Random Forest:
El segundo en performance.
max_depth=10, min_samples_leaf=15, n_estimators=1000, score=0.86320

Red Neuronal:

Fue el 3 en performance de las predicción de cualquier forma no fueran nada malas, decimos usarlo por que suele dar buenos resultados,activation='relu',solver='lbfgs', alpha=1e-5,hidden_layer_sizes=15, random_state=5 el score en base a AUC fue score=0.82258

Knn:

Nos entregó las predicciones más bajas, utilizando un grid search y obtuvimos con k= 150 resultados en base a la medida AUC de score= 0.77

Conclusiones:

Mientras realizamos el tp fuimos descubriendo cosas interesantes acerca de los datos, inicialmente no realizamos features temporales y las predicciones no rankeaban del todo bien, al agregar features temporales el tp comenzó a mejorar notablemente, esto nos mostró que realizar ese tipo de features tiene gran peso en este tp, entre las columnas que más importancia tuvieron fueron, el tiempo que pasó desde el último evento hasta la fecha límite, y las distancias de tiempo hasta la fecha límite de varios de los eventos.

los datos provistos para el tp fueron entregados por:https://www.trocafone.com.ar/
links
https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/

link al git: https://github.com/villanicr/tp2-datos