UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

75.06 Organización de Datos

Trabajo Práctico N°2

Segundo Cuatrimestre de 2018

Grupo 36

Franchi, Eugenia	100855
Sanjines, Aaron	99176
Zugna, Federico	95758

Link de GitHub: https://github.com/EugeniaFranchi/TP-Datos/tree/master/TP2

Contenido

INTRODUCCIÓN

En este trabajo practico intentaremos predecir la probabilidad de que un usuario compre (realice una conversion). Para esto realizaremos un análisis de los posibles features que mediante algoritmos de Machine Learning permitan acertar lo máximo posible esta probabilidad.

ENUNCIADO DEL TRABAJO PRACTICO

El segundo trabajo práctico es una competencia de Machine Learning en donde cada grupo debe intentar determinar, para cada usuario presentado, cuál es la probabilidad de que ese usuario realice una conversión en Trocafone en un periodo determinado.

La competencia se desarrollará en la plataforma de Kaggle. En el siguiente link se provee la siguiente información para la competencia:
https://drive.google.com/file/d/1kQujhvOKAU4EzhaYDbgquf_XKFt9sHMy/view?usp=sharing

En este podrán encontrar los siguientes archivos:

```
events_up_to_01062018.csv
labels_training_set.csv
```

El archivo events_up_to_01062018.csv contiene en el mismo formato utilizado en el TP1 información de eventos realizado en la plataforma para un conjunto de usuarios hasta el 31/05/2018.

Por otro lado el archivo labels_training_set.csv indica para un subconjunto de los usuarios incluidos en el set de eventos events_up_to_01062018.csv si los mismos realizaron una conversión (columna label = 1) o no (columna label = 0) desde el 01/06/2018 hasta el 15/06/2018.

```
conversiones_train_pub =
pd.read_csv('Data/Trocafone/Pub/labels_training_set.csv')
conversiones_train_pub.head()
```

	person	label
0	0566e9c1	0
1	6ec7ee77	0
2	abe7a2fb	0
3	34728364	0

4	87ed62de	0

La información de estos archivos debe ser utilizada para entrenar un modelo de Machine Learning, de tal forma de poder indicar la probabilidad de que conjunto seleccionado de usuarios realice una conversión desde el 01/06/2018 al 15/06/2018.

Se pedirá indicar esa probabilidad de conversión para usuarios que no se encuentran en el archivo `labels_training_set.csv`, pero para los cuales se cuenta con información en `events_up_to_01062018.csv`

El listado de estas personas será provisto en el archivo trocafone_kaggle_test.csv

```
personas_a_predecir =
pd.read_csv('Data/Trocafone/trocafone_kaggle_test.csv')
personas_a_predecir.head()
```

	person
0	4886f805
1	0297fc1e
2	2d681dd8
3	cccea85e
4	4c8a8b93

El link a la competencia es https://www.kaggle.com/t/f477d6cc8ce34161a33bcc02ad055912

Los grupos deberán probar distintos algoritmos de Machine Learning para predecir cuál es la probabilidad de conversión del conjunto de usuarios seleccionados de Trocafone para la competencia en el periodo descrito. A medida que los grupos realicen pruebas deben realizar el correspondiente submit en Kaggle para evaluar el resultado de los mismos.

FEATURE ENGINEERING

A continuación, daremos detalles de los features hallados en la resolución del trabajo practico.

Una medida tomada fue la de agrupar los eventos realizados según el par de semanas en que tuvieron lugar. Es decir, dividimos los 6 meses (que es el periodo en el que transcurrieron los datos provistos) en grupos de dos semanas ya que el objetivo es predecir la compra en un periodo de dos semanas. Llamaremos el intervalo de dos semanas 'periodo' para mayor fluidez en el desarrollo.

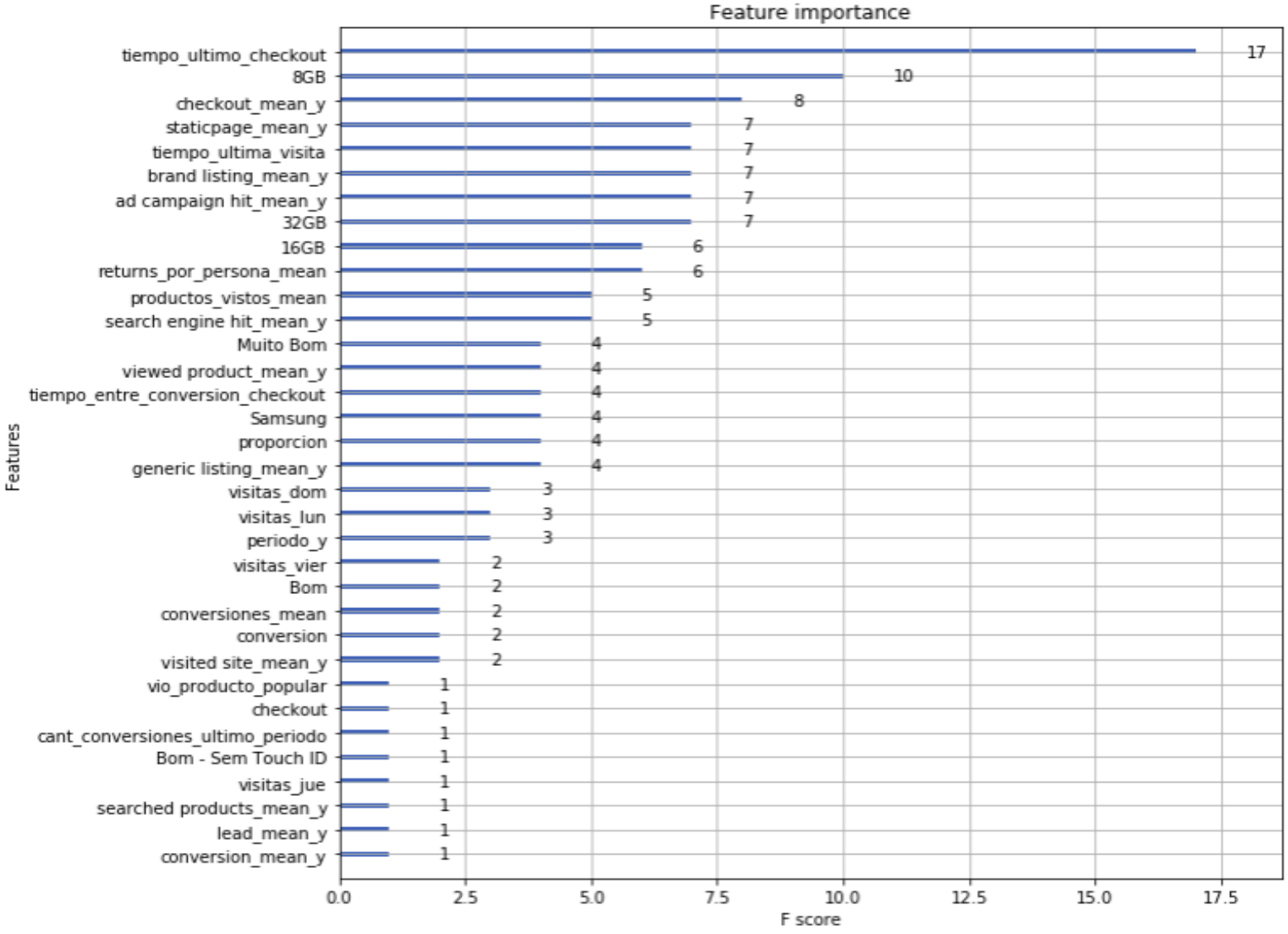

Figura 1: Gráfico de la importancia de cada feature.

En este cuadro [Figura 1] observamos la importancia de cada feature en el algoritmo con el que mejor resultado obtuvimos: XGBoost

Promedio de eventos realizados en un periodo

- 'ad campaign hit_mean_y'
- 'brand listing_mean_y'
- 'checkout_mean_y',
- 'conversion_mean_y'
- 'generic listing_mean_y'
- 'lead_mean_y',
- 'search engine hit_mean_y'
- 'searched products_mean_y',
- 'staticpage_mean_y'
- 'viewed product_mean_y'
- 'visited site_mean_y'

Alguno de estos features tenían gran impacto al momento de calcular la probabilidad de la conversion, por ejemplo ‘checkout_mean’, pero también había algunos que no solo no sumaban, sino que por el contrario hasta empeoraban el score, así como ‘conversion_mean’.
Una observación curiosa e irónica es que uno pensaría que la cantidad de conversiones realizadas seria de los más importantes o con más valor, pero el algoritmo de predicción utilizado no le dio mucha importancia [Figura 1].

Promedio de eventos que involucran diferentes tamaños de memoria en dispositivo por periodo

- '4GB'
- '8GB',
- '16GB',
- '32GB'
- '64GB'
- '128GB'
- '256GB'
- '512MB'

Por el cuadro de importancia, vimos que tres de estos (8GB, 32GB, 16GB) se encuentran en el top 10 de features más trascendentes.

Observamos por un lado que los dispositivos que cuentan con 32GB y 16GB de almacenamiento son los que más apariciones tuvieron en eventos. Pensamos que esto es un motivo clave para su importancia, aunque no sabemos explicar por qué 8GB tiene más importancia que 32GB y 16GB (y que las demás memorias inclusive) teniendo en cuenta que tiene menos apariciones que estos. Dicha singularidad acaba con la teoría propuesta que decía que la importancia del feature estaba relacionada con la cantidad de apariciones.

Features relacionados con el tiempo

- 'tiempo_ultima_conversion'
- 'tiempo_ultimo_checkout'
- 'tiempo_ultima_visita'
- 'tiempo_ultimo_lead'
- 'tiempo_entre_checkout_mean_log'
- 'tiempo_entre_conversion_checkout'
- 'dias_desde_ultimo_evento'

Al igual que el feature del almacenamiento, los features relacionados con el tiempo son de los que mejor resultados mostraron, como podemos ver en el gráfico. En primero lugar encontramos al que muestra, no solo una diferencia significativa con respecto a las demás, sino que además es el que tiene mayor importancia: tiempo_ultimo_checkout.

También podemos observar que los otros dos features con esta temática, tiempo_entre_conversion_checkout y tiempo_ultima_visita, que aparecen en el gráfico tienen un posicionamiento relativamente bueno comparado con el promedio (están de mitad de tabla para arriba)

Features relacionados a preguntas cerradas

- 'vio_mas_de_5_veces'
- 'vio_color_mas_vendido'
- 'vio_producto_popular’

De estos features, el algoritmo solo tuvo en cuenta (por el learning_rate) a 'vio_producto_popular’ aunque de todas formas no tuvo gran repercusión a la hora de predecir.

ALGORITMOS UTILIZADOS

KNN

Este fue el que peor resultado dio. Aunque tuvo un gran porcentaje de acierto para predecir la clase, no lo tuvo así para la probabilidad de que un usuario pertenezca a la clase 1
Algoritmo con entrenamiento barato, pero no es el mejor para predecir

MAXIMO SCORE DE TEST = %75

Random Forest

Tuvo un gran acierto, pero lamentablemente tampoco fue el mejor. Supo corregir los errores de predicción de probabilidad realizados por KNN y hacer que veamos llamativas mejoras en el score de Kaggle. Aunque se lo considera un algoritmo de ensamble, decidimos tomarlo en cuenta como un algoritmo aparte ya que prometía por sí mismo grandes resultados (y no defraudo del todo)

MAXIMO SCORE DE TEST = %83

XGBoost

El mejor algoritmo “individual” probado por el grupo. Nos metió y mantuvo en los puestos del top 15 de la competencia.
Era de esperar ya que este algoritmo es “el estado del arte” en clasificación. Luego de las optimizaciones saco una pequeña diferencia numérica, pero grande en resultado, de los demás algoritmos

MAXIMO SCORE DE TEST = %87

ENSAMBLES

Voting

Primer ensamble probado del grupo de algoritmos de ensamble.
Dio mejores resultados que KNN y Random Forest, aunque no puedo superar a la bestia XGBoost. Pudo haber sido un gran incentivo al principio del trabajo práctico ya que mejoró notablemente los resultados de score

MAXIMO SCORE DE TEST = %85

Bagging

Bagging del algoritmo XGBoost resulto ser el algoritmo elegido por el grupo. Esta técnica de ensamble hizo que mejorara la efectividad del score usando los mismos features que los demás algoritmos. Con alguna mejora y agregado de features pudo tranquilamente haber ganado la competencia.

MAXIMO SCORE DE TEST = %89

AdaBoost

Algoritmo de boosting utilizado con XGBoost. Genero un gran resultado superando a todos los demás algoritmos, incluyendo a Voting, pero no tuvo la capacidad de superar el Bagging de XGBoost. Igualmente, es una gran opción para predecir

MAXIMO SCORE DE TEST = %88

PROCESAMIENTO DE DATOS

Hemos aplicado dos técnicas al set de entrenamiento para evitar el overfitting:

Cross-Validation

Hemos buscado los mejores hiperparámetros mediante el módulo de Sklearn 'GridSearchCV'.

K-Fold

Con K-Fold separamos el set de entrenamiento en 10 bloques y luego de una evaluación, nos quedamos con el bloque optimo, es decir, con el que no dio mejor precisión el algoritmo

CONCLUSIONES

- La más obvia quizás: El ensamble de algoritmos funciona mejor que los algoritmos por separado

- Conviene que los features contengan cierta relación para que el algoritmo aproxime mejor. Por ejemplo, cuando usábamos features relacionados con el promedio de eventos y agregamos un nuevo feature que indica si el usuario vio un dispositivo de color oro, el algoritmo perdía precisión

- XGBoost es claramente el mejor algoritmo de clasificación y esto es por la gran precisión que puede conseguir

- Utilizar el promedio para codificar features categóricos da muy buenos resultados