Universidad de Buenos Aires

Facultad de Ingeniería

(75.06) Organización de Datos

Cátedra: Luis Argerich

Repositorio: https://github.com/freedocx/Trocafone-Analysis

2° Cuatrimestre 2018

Alumnos:

GARATE, Julián Matías (93043)

Materia: 75.06 Organización de Datos		Repo: https://github.com/freedocx/Trocafone-Analysis
Predicción	Cuatrimestre: 2C 2018	Probabilidad de Conversión

Contenido

Materia: 75.06 Organización de Datos		Repo: https://github.com/freedocx/Trocafone-Analysis
Predicción	Cuatrimestre: 2C 2018	Probabilidad de Conversión

Introducción

La empresa Trocafone tiene un servicio de re-commerce de electrónica. Adquieren celulares y tablets usados, los acondicionan, los venden con garantía y múltiples opciones de pago.
Este trabajo se trata de una competencia de Machine Learning para predecir compras futuras de usuarios en base a su comportamiento presente y pasado.
Link: https://www.trocafone.com.ar/

Análisis del set de Datos

El archivo `events_up_to_01062018.csv` contiene los datos de los eventos que se registraron entre 2018-01-01 08:09:31 y 2018-05-31 23:59:59. Cada eventos está asignado a un usuario correspondiente
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.
Por último un archivo `trocafone_kaggle_test.csv contiene un` subconjunto de los usuarios incluidos en el set de eventos `events_up_to_01062018.csv para los cuales se deben realizar las predicciones.`

Lenguaje de Programación

Para este trabajo, se utilizó Python 3, sobre la plataforma de Anaconda y las siguientes librerías

- Sklearn: brinda todos los algoritmos para el procesamiento y el formato de datos
- Pandas: lectura y trabajo de análisis sobre las estructuras de datos
- Xgboost: algoritmo de machine learning

Selección del Algoritmo

Para la primera prueba simplemente se eligió hacer una Regresión Lineal, con el nivel de actividad para cada usuario, manifestado en cada uno de los distintos eventos. El resultado en pruebas internas fue del 78%, esto muestra la gran relevancia que tiene la actividad del usuario para predecir su comportamiento futuro.
Al sumar features con las preferencias de los usuarios en tipo de dispositivo, estado del celular/tablet, etc, se cambió por XGBoost con el cual se logró un 87.4%.

XGBoost

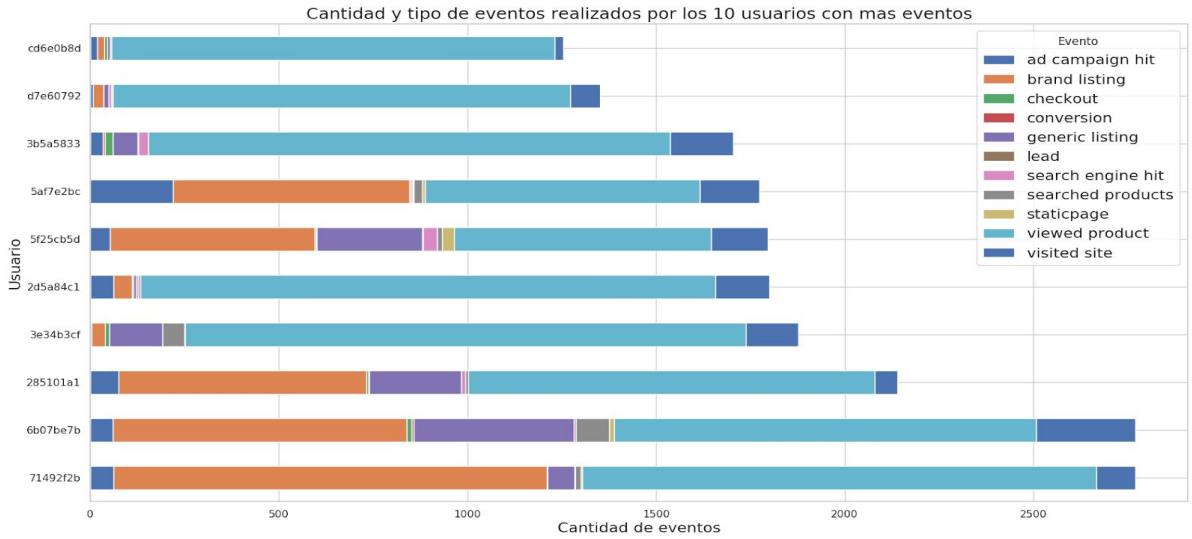
Es una aplicación de árboles de decisión que utiliza el método de Boosting. Sencillamente el algoritmo se compone de un una función error, una función de regularización para mantener conservador el modelo (y controlar la complejidad de los árboles) y un conjunto de árboles anidados los cuales trabajan sobre el error del anterior. El objetivo del algoritmo es minimizar la suma del error con la regularización, de esa manera el modelo resultante evita el sobreajuste, derivado de una excesiva complejidad, mientras decrece el error.
En este caso el ajuste de los hiper parámetros, pasa por ajustar la función de regularización, la tasa de aprendizaje (ganancia), el tamaño y profundidad de los árboles.

Desarrollo

Preparación de los features

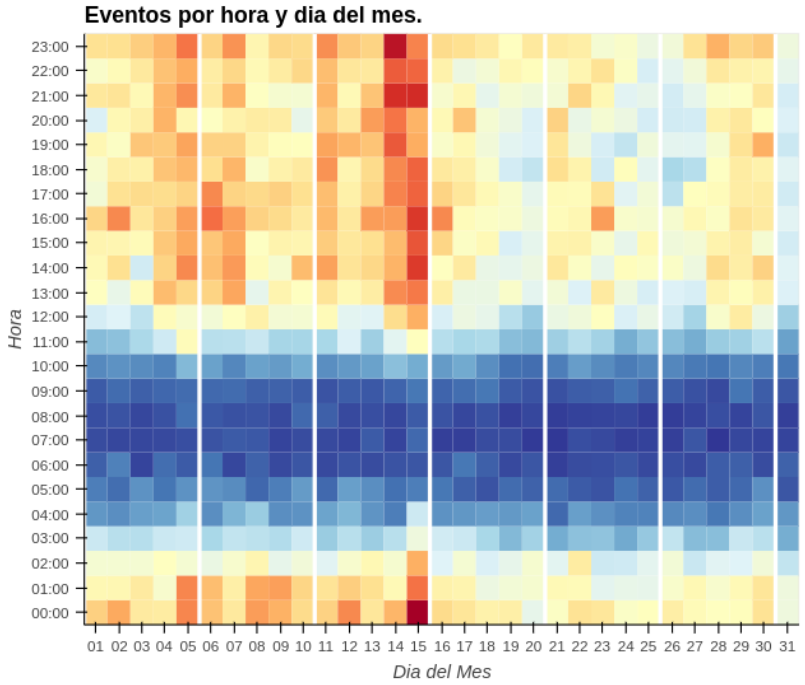
Cantidad de eventos, para determinar nivel de actividad

1. Cantidad de Eventos totales de una persona, lo mismo para cada uno de los distintos tipos de eventos. Hay algunos genéricos, como la visita, pero hay otros determinantes, como la cantidad de checkout y conversiones a pesar de ser pocos.

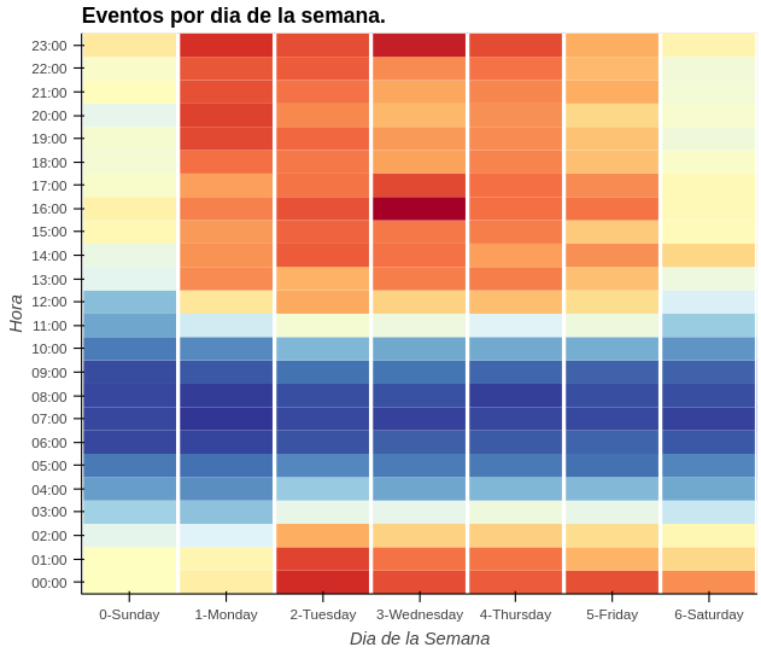


2. Dado que se predicen los 15 días posteriores al último evento registrado en los datos, se contaron los eventos y los tipos de eventos de los usuarios en los **últimos días al cierre del mes de mayo.** Discriminando también la primera de la segunda quincena del mes (cantidad del 1 al 15 y del 16 al 31 de mayo).

Fechas: como se puede apreciar en el heatmap, se arman franjas tanto horarias como de días por la diferencia en la cantidad de eventos.



1. Se armaron franjas horarias contando los eventos entre **0-7, 8-15,16-23 hs**.
2. Se contaron los eventos para **cada día del mes**, para todo el periodo desde enero.
3. Se contaron los eventos para **cada día de la semana**, para todo el periodo desde enero.



Condición de los productos: hay usuarios que buscan productos de diferentes estados y calidades. Se arman columnas donde se cuentan los eventos para cada tipo de condición de cada usuario.
Ciudad: los usuarios realizan los eventos desde distintas localidades, se busco la localidad principal de cada usuario y se calculó para cada localidad una relacion eventos/conversion

```
TasaConversionXciudad = (conversionXciudad/eventoXciudad).fillna(0)
TasaConversionXciudad = TasaConversionXciudad.reset_index()
```

Modelos: de la misma manera que para la ciudad, se busca el modelo más representativo para cada usuario y se le calcula una relación entre eventos/conversion

```
TasaConversionXmodelo = (conversionXmodelo/eventoXmodelo).fillna(0)
TasaConversionXmodelo = TasaConversionXmodelo.reset_index()
```

Storage: para armar este feature, se calculó el promedio de capacidad buscada de cada usuario.

```
eventos[['person','storage']].groupby('person').sum().reset_index()
```

Device Type: para cada tipo de dispositivo, al ser pocos, se los transforma en columnas y se cuenta la cantidad de eventos para cada uno, hecho por cada usuario.

Sistema operativo: para este feature se armó una relación como la de para cada modelo.

Marcas: se deducen de los modelos

Calificación por usuario: basado en lo visto en el análisis exploratorio, se tiende a producir un flujo de navegación que va desde la entrada del sitio hasta una conversión (si es que se produce). Teniendo en cuenta la retroalimentación de algunos eventos se armó una métrica para comparar a los usuarios bajo el siguiente criterio, que dio el mejor resultado:

```
eventos['event'].loc[eventos['event'] == 'visited site'] = 0
eventos['event'].loc[eventos['event'] == 'search engine hit'] = 0
eventos['event'].loc[eventos['event'] == 'ad campaign hit'] = 1
eventos['event'].loc[eventos['event'] == 'staticpage'] = 0
eventos['event'].loc[eventos['event'] == 'generic listing'] = 2
eventos['event'].loc[eventos['event'] == 'searched products']= 1
eventos['event'].loc[eventos['event'] == 'brand listing'] = 2
eventos['event'].loc[eventos['event'] == 'viewed product'] = 3
eventos['event'].loc[eventos['event'] == 'checkout']= 3
eventos['event'].loc[eventos['event'] == 'conversion'] = 8
eventos['event'].loc[eventos['event'] == 'lead']= 8
```

Escalar

Las pruebas fueron entre StandardScaler y MaxMinScaler de **sklearn**.
MaxMinScaler dio mejor resultado.

```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
```

El resultado es un dataset con una escala unificada para todas las variables.

CrossValidation-GridSearch

```
X_train, X_test, y_train, y_test = train_test_split(train, train_y, test_size=0.5, random_state=2354)
y_test.shape , X_test.shape , X_train.shape, y_train.shape
```

Se hizo una separación para hacer una validación de última instancia antes de subir resultados a Kaggle

La elección de los hiper parámetros del algoritmo se automatizó con Cross Validation y Grid Search. El cálculo del error se determinó, como en la competencia por el área por debajo de la curva (**auc**). Las siguientes combinaciones fueron probadas.

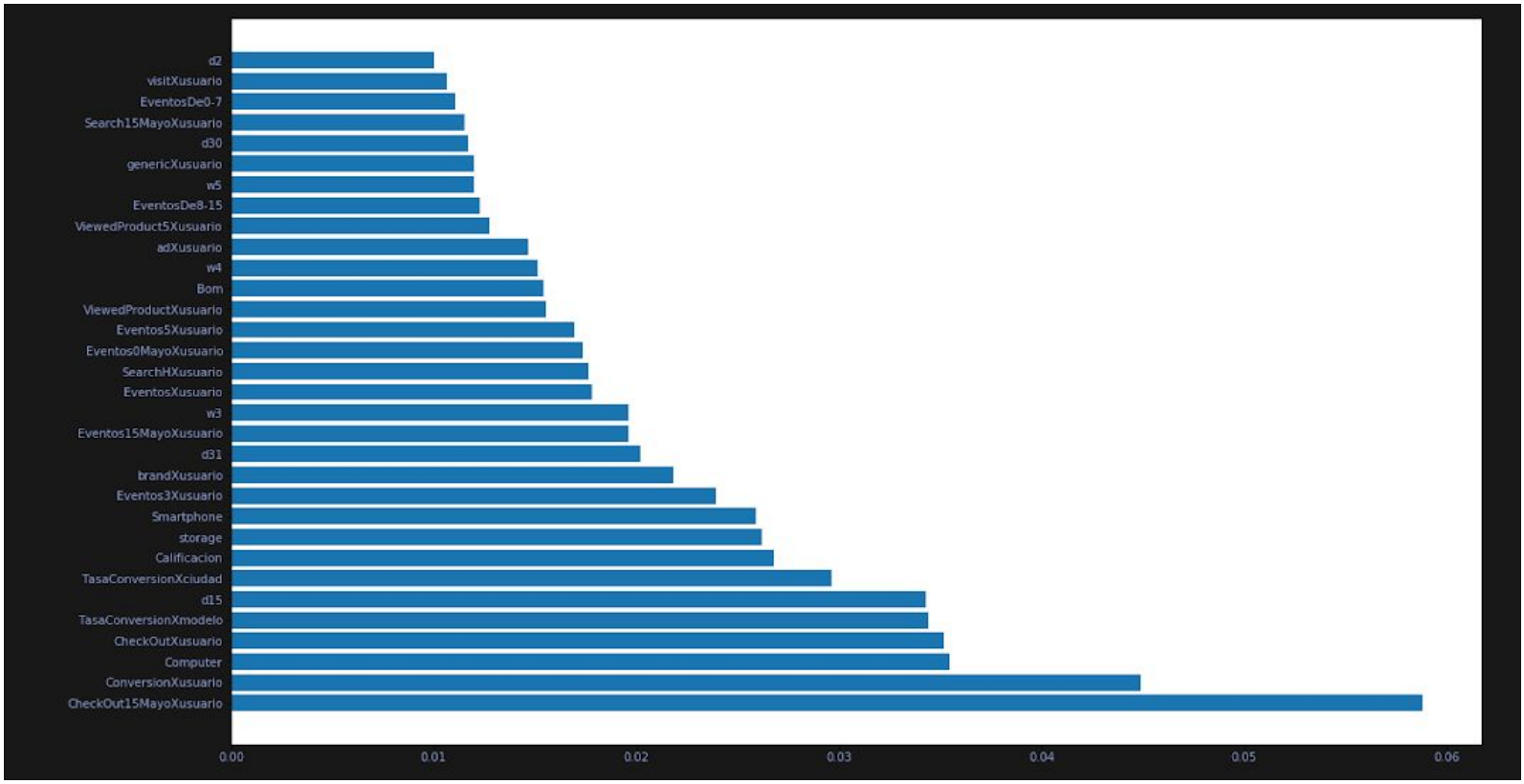
```
params = {'learning_rate': [0.01,0.02,0.05,0.1,0.005],
    'max_depth': [5],
    'gamma': [0,1,2,4,5],
    'n_estimators': [100,200,300,400,500,600],
        'colsample_bylevel':[0.2,0.3,0.4,0.5,0.6,0.7,0.8]}
xgb_model = xgb.XGBClassifier(eval_metric='auc',n_jobs=4, early_stopping_rounds=15,silent=True)
clf = GridSearchCV(xgb_model,params,scoring ='roc_auc',cv=5)
#clf.fit(X_train,y_train)
clf.fit(train,train_y)
```

Ajustando el la tasa de aprendizaje, la regularización para aportar conservadurismo, la profundidad de los árboles, la cantidad de columnas, y el número.

```
results.sort_values(by='mean_test_score',ascending=False)[0:5]
```

mean_test_score	mean_train_score	param_colsample_bylevel	param_gamma	param_learning_rate	param_max_depth	param_n_estimators
0.873336	0.941271	0.4	5	0.02	5	500
0.873273	0.938311	0.4	5	0.02	5	400
0.873239	0.932703	0.4	5	0.02	5	300
0.873228	0.943315	0.4	5	0.02	5	600
0.873217	0.948598	0.4	1	0.02	5	300

Importancias de los features para el entrenamiento del algoritmo



Claramente se puede ver el lugar que le asigna a cada feature el algoritmo. De este entrenamiento se logra el 87.4%.
Del análisis completo tanto exploratorio como de preparación para entrenar el algoritmo y de la salida de XGBoost se llega a las conclusiones.

Conclusiones

#	Tema	Conclusion
1	Nivel de Actividad	El nivel de actividad, medido por la cantidad de eventos y tipos da un colchón de probabilidades de 85% para predecir una compra. A destacar ❑ **Cantidad de Eventos** ❑ **Cantidad de CheckOuts en los últimos 15 días.** ❑ **Cantidad de Conversiones por usuario.** ❑ **Cantidad de CheckOuts**. ❑ Cantidad de **BrandListing**
2	Fechas	Las fechas resultaron ser muy relevantes para predecir comportamiento de los usuarios:: ❑ Tanto el 31,30 tienen actividad relevante. Se podría asumir que los que tienden a **mirar productos a fin de mes, tienden a concretar una compra en el transcurso de la quincena siguiente**. ❑ **El 15 es relevante por el nivel de actividad** que se aprecia en el heatmap. ❑ El 2 y el 3 son días de relativa importancia para el algoritmo, posiblemente porque para esas fechas ya se empieza a percibir el sueldo.
3	Horario	Las franjas horarias de la tarde y de la madrugada resultaron ser más útiles para diferenciar a los usuarios.
4	Semana	En los días de la semana el miércoles es el que más se destaca, posiblemente por su alto nivel de actividad.:
5	Calificación	**No todos los usuarios tienden a concretar compras** después de un ciclo de navegación, el asignarles puntaje, le da al algoritmo de ML la posibilidad de discriminarlos por el tipo de navegación que hacen en el sitio.
6	Dispositivo de entrada	Hay mayor cantidad de eventos por Mobile pero hay mayor cantidad de conversiones en Desktop. **Pareciera que las personas tienden a concretar sus compras en una computadora de escritorio.**
7	Storage	El storage es una variable relevante para predecir en base a las capacidades que buscan los usuarios.

Bibliografía

- https://xgboost.readthedocs.io/en/latest/
- http://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#sphx-glr-auto-examples-preprocessing-plot-all-scaling-py
- https://pandas.pydata.org/
- http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- http://scikit-learn.org/stable/modules/cross_validation.html
- http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.