FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2: Machine Learning

[75.06 / 95.58] Organización de Datos
Segundo cuatrimestre de 2018

Grupo Datatouille

Alumno	Padrón	Mail
Bojman, Camila	101055	camiboj@gmail.com
del Mazo, Federico	100029	delmazofederico@gmail.com
Hortas, Cecilia	100687	ceci.hortas@gmail.com
Souto, Rodrigo	97649	rnsoutob@gmail.com

https://github.com/FdelMazo/7506-Datos/
https://kaggle.com/datatouille2018/

Curso 01

- Argerich, Luis Argerich
- Golmar, Natalia
- Martinelli, Damina Ariel
- Ramos Mejia, Martín Gabriel

Índice

1. Introducción

El objetivo principal del trabajo es predecir la probabilidad de que un usuario de la empresa Trocafone realice una compra de un dispositivo celular (conversión).

La realización del trabajo se hace con algoritmos de Machine Learning, una disciplina que busca poder generar clasificaciones en base a un entrenamiento sobre información pasada, seguida de una validación de las predicciones generadas. En el trabajo se prueban distintos algoritmos, los cuales todos en distinta manera hacen uso de los datos (en particular, de sus atributos). Es por esto que es muy importante saber que datos usar, y buscar como codificarlos de tal forma que mejor se aprovechen.

Con dicho propósito se utilizan como base dos sets de datos brindados por la empresa. En un primer lugar el archivo `events_up_to_01062018.csv` que contiene la información de los eventos realizados por un conjunto de usuarios desde el 1ro de enero hasta el 31 de mayo de 2018 y servirá como entrenamiento de los algoritmos, y en un segundo lugar el archivo `labels_training_set.csv` que determina si el usuario realizó o no una conversión desde el 1ro hasta el 15 de junio es este período es del cual se quieren predecir las conversiones). Siendo este segundo archivo un subconjunto del anterior, este servirá de validación de las predicciones

2. Organización del Trabajo

Para un desarrollo más cómodo, se modularizó el trabajo en distintos notebooks de Jupyter, intentando emular la programación estructurada, y luego se importan entre si [1]. Si bien esta forma de trabajar sirve bastante para no estar repitiendo código, el dividir en distintos notebooks que dependan unos de otros sube el acoplamiento del proyecto en su totalidad.

Los notebooks, en orden de lectura y corrida son:

1. **Investigación Previa**

   https://fdelmazo.github.io/7506-Datos/TP2/investigacion.html

   En este notebook se presentan las distintas exploraciones que se hicieron sobre el dataset del trabajo práctico anterior (TP1), en busqueda de ideas útiles para el desarrollo del actual.

2. **Creación de dataframes**

   https://fdelmazo.github.io/7506-Datos/TP2/new_dataframes.html

   En este notebook se crean los distintos dataframes necesarios para poder extraer atributos de los usuarios.

3. **Feature engineering**

   https://fdelmazo.github.io/7506-Datos/TP2/feature_engineering.html

   En este notebook se agregan todos los features que se consideran que pueden ser pertinentes para el modelo. Este notebook es el que genera el `user-features.csv` del cual entrenan los modelos.

[1] Haciendo uso del magnífico nbimporter

4. **Feature selection**

   https://fdelmazo.github.io/7506-Datos/TP2/feature_selection.html

   En este notebook se utilizan distintas formas de seleccionar los features con el objetivo de eliminar ruido y encontrar la mejor combinación de atributos a usar a la hora de entrenar modelos.

5. **Parameter tuning**

   https://fdelmazo.github.io/7506-Datos/TP2/parameter_tuning.html

   En este notebook se hacen diversas pruebas sobre cada algoritmo hasta encontrar los hiper parametros óptimos de cada uno.

6. **Submission framework**

   https://fdelmazo.github.io/7506-Datos/TP2/submission_framework.html

   La principal idea de este notebook es definir una serie de pasos para armar las postulaciones de predicciones del trabajo práctico.

7. **Notebook principal**

   https://fdelmazo.github.io/7506-Datos/TP2/TP2.html

   Finalmente, haciendo uso del framework previamente definido, se encuentra la combinación óptima de atributos y algoritmos para hacer una postulación de predicciones.

3. Features

3.1. Investigación previa

El primer paso del trabajo consistió en realizar una investigación sobre lo ya hecho en el trabajo anterior. El TP1 [2] es un análisis exploratorio de datos de la empresa. Si bien no son exactamente los mismos datos que los trabajados acá, son de la misma índole, y la exploración de ellos dan a luz a patrones en los usuarios del sitio.

Esta investigación se compone de dos partes, una técnica y otra teórica.

Por el lado técnico, viendo que se uso otro set de datos para el TP1, se buscó alguna forma de integrar los datos anteriores con los nuevos (por ejemplo, buscar si hay usuarios compartidos entre los dos sets, o si hay compras para registrar), para poder usar una base de datos más grande tanto para el entrenamiento como la validación de las predicciones.

Luego de una serie de pasos, busquedas y validaciones, se recopiló la siguiente información:

1. No se repiten usuarios en los datasets.

2. En el primer dataset (TP1) hay 27624 usuarios de los cuales 13967 tuvieron actividad en junio. Entre el 1 y el 15 (inclusive) de junio 82 usuarios compraron productos.

3. En el segundo dataset hay 19414 usuarios de los cuales 980 compraron en Junio.

[2] https://fdelmazo.github.io/7506-Datos/TP1/TP1.html

Por lo tanto se concluyó que hacer un merge de los datos del TP1 con los del TP2 presentaría un *skewness* en el set de datos, por la despreciabilidad de estos.

Por otro lado, en un marco teórico, se vió el análisis hecho en busqueda de que patrones, ideas y conceptos pueden ser aplicados en este trabajo. En particular, se buscan atributos escondidos en el set original que puedan ser codificados de tal forma que luego los algoritmos de Machine Learning puedan utilizar a su favor. Los atributos encontrados son especificados en la sección de Feature Engineering.

[TO-DO TSNE de Souto aca]

3.2. Creación de dataframes

Este es mayoritariamente un re-trabajo sobre lo hecho para el TP1, en el Notebook Anexo[3].

Como parte del feature engineering, se crean dataframes nuevos con información de los productos del sitio y de como se accede a este. Los dataframes generados son:

- `brands.csv`: Lista las marcas de los dispotivos de cada evento.
- `os.csv`: Lista los sistemas operativos desde los cuales se accedió al sitio.
- `browsers.csv`: Lista los exploradores desde los cuales se accedió al sitio.
- `sessions.csv`: Se agregó el concepto de sesión, que se define como la agrupación de una serie de eventos por usuario, los cuales están todos con menos de 30 minutos de inactividad entre el actual y el anterior.
- `prices.csv`: Lista los precios de los dispositivos de cada evento. Para lograr esto se hizo un *web-scraping* de la página de Trocafone de la cual se extrajo para cada conjunto de modelo, capacidad, color y condición el precio del dispositivo.

3.3. Feature engineering

Con lo investigado del previo trabajo y todos los dataframes generados, se busca todo tipo de atributos de los usuarios, para que luego puedan ser seleccionados y aprovechados por los algoritmos a aplicar.

3.3.1. Suma total de eventos

```
['total_viewed_products', 'total_checkouts',
'total_conversions', 'total_events', 'total_sessions',
'total_session_checkout', 'total_session_conversion',
'total_events_ad_session', 'total_ad_sessions',
'avg_events_per_session', 'avg_events_per_ad_session',
'percentage_session_ad', 'percentage_session_conversion',
'has_checkout', 'has_conversion']
```

Los atributos refieren a la sumatoria de cantidad de productos vistos, en checkout y en coversiones, más la totalidad de los eventos de cada usuario. Lo

[3] https://fdelmazo.github.io/7506-Datos/TP1/anexo.html

mismo para las sesiones de cada usuario y sus subtotales. También se agregan promedios de eventos por sesión y porcentajes de los accesos de las sesiones del usuario.

3.3.2. Cantidad de eventos por mes

['total_viewed_products_month_1', 'total_checkouts_month_1',
'total_conversions_month_1', 'total_events_month_1',
'total_sessions_month_1', 'total_session_checkouts_month_1',
'total_session_conversions_month_1', 'total_events_ad_session_month_1',
'total_ad_sessions_month_1', 'has_checkout_month_1',
'has_conversion_month_1', 'total_viewed_products_month_2',
'total_checkouts_month_2', 'total_conversions_month_2',
'total_events_month_2', 'total_sessions_month_2',
'total_session_checkouts_month_2', 'total_session_conversions_month_2',
'total_events_ad_session_month_2', 'total_ad_sessions_month_2',
'has_checkout_month_2', 'has_conversion_month_2',
'total_viewed_products_month_3', 'total_checkouts_month_3',
'total_conversions_month_3', 'total_events_month_3',
'total_sessions_month_3', 'total_session_checkouts_month_3',
'total_session_conversions_month_3', 'total_events_ad_session_month_3',
'total_ad_sessions_month_3', 'has_checkout_month_3',
'has_conversion_month_3', 'total_viewed_products_month_4',
'total_checkouts_month_4', 'total_conversions_month_4',
'total_events_month_4', 'total_sessions_month_4',
'total_session_checkouts_month_4', 'total_session_conversions_month_4',
'total_events_ad_session_month_4', 'total_ad_sessions_month_4',
'has_checkout_month_4', 'has_conversion_month_4',
'total_viewed_products_month_5', 'total_checkouts_month_5',
'total_conversions_month_5', 'total_events_month_5',
'total_sessions_month_5', 'total_session_checkouts_month_5',
'total_session_conversions_month_5', 'total_events_ad_session_month_5',
'total_ad_sessions_month_5', 'has_checkout_month_5',
'has_conversion_month_5']

[TO-DO: Reemplazar texto de abajo por algo mas generico]

Se agregan una serie de features relacionados a la cantidad de eventos y sesiones por mes que se consideraron pertinentes al modelo.

- `total_viewed_products_month`: cantidad de productos vistos por mes por usuario
- `total_checkouts_month`: cantidad de productos que llegaron a checkout por mes por usuario
- `total_conversions_month`: cantidad de productos que llegaron a ser comprados por mes por usuario
- `total_events_month`: cantidad de eventos por mes por usuario
- `total_sessions_month_`: cantidad total de sesiones por mes
- `total_session_checkouts_month_`: cantidad total de sesiones donde el usuario hace checkout por mes

- `total_session_conversions_month_`: cantidad total de sesiones donde el usuario compra un producto por mes
- `total_events_ad_session_month_`: cantidad total de sesiones donde el usuario ingresa a la página por una campaña publicitaria por mes
- `total_ad_sessions_month_`: cantidad total de sesiones donde el usuario ingresa a la página por primera vez por una campaña publicitaria por mes

3.3.3. Eventos sin contar mayo

3.3.4. Eventos en última semana

3.3.5. Distribución mensual de las conversiones

Se agrega en cuántos meses el usuario compró suponiendo que dicha distribución denota si el usuario es un comprador habitual o sólo compró alguna vez aisladamente. El feature se llama .ªmount_of_months_that_have_bought".

3.3.6. Informacion de los últimos eventos registrados por usuario

Se busca extraer información de los días que transcurrieron hasta el último evento de un usuario. De esta manera se espera que el modelo aprenda un factor importante para la predicción. Por ejemplo, si un usuario vio un producto hace muchos días es muy probable que no lo compre pero si hizo checkout hace 1 dia es probable que en un futuro cercano compre.

- `days_to_last_event`: cantidad de días hasta el último evento
- `days_to_last_checkout`: cantidad de días hasta el último checkout. Si el usuario no hizo checkout se considera un número mayor a la cantidad de días del período de tiempo comprendido.
- `days_to_last_conversion`: cantidad de días hasta la última compra del usuario. Si el usuario nunca compró se considera un número mayor a la cantidad de días del período de tiempo comprendido.
- `days_to_last_viewed_product`: cantidad de días hasta el último día que el usuario vio un producto. Si el usuario nunca vio un producto se considera un número mayor a la cantidad de días del período de tiempo comprendido.

En paralelo con estos features se consideran los días de la semana, del mes, del año y la semana del año donde ocurren estos últimos eventos.

3.3.7. Precios de la ultima conversion realizada por el usuario

Se consideró que podría considerarse el precio de la última conversión del usuario como un feature pero a la hora de la selección reflejó una importancia muy baja. Por lo tanto consideramos impertinente la descripción de la idea que habíamos pensado desarrollar.

3.3.8. Porcentaje de la actividad de la ultima semana

Aquí la idea pensada era reflejar la cantidad de eventos del usuario de la última semana sobre el total. Si el usuario ingresó muchas veces a la página en la última semana de mayo es muy probable que compre en la primera semana de junio. De la misma manera, si el usuario compró la última semana de mayo es probable que no compre por las siguientes dos.

Por lo tanto se pensaron los siguientes features:

- `percentage_last_week_activity`: porcentaje de la cantidad de eventos de esa semana sobre el total de eventos
- `percentage_last_week_conversions`: porcentaje de la cantidad de compras de esa semana sobre el total de eventos
- `percentage_last_week_checkouts`: porcentaje de la cantidad de checkouts de esa semana sobre el total de eventos
- `percentage_last_week_viewed_products`: porcentaje de la cantidad de productos vistos de esa semana sobre el total de eventos

3.3.9. Porcentaje de la actividad del ultimo mes

Una lógica análoga a la sección precedente se sigue en esta parte. Los motivos de este feature son simplemente una ampliación de la idea anterior. Si el usuario ingresó muchas veces a la página en mayo es muy probable que compre en la primera semana de junio. De la misma manera, si el usuario compró en mayo es algo probable que no compre por las siguientes dos.

De más está decir que se pensaron los siguientes features:

- `percentage_last_month_activity`: porcentaje de la cantidad de eventos de ese mes sobre el total de eventos
- `percentage_last_month_conversions`: porcentaje de la cantidad de compras de ese mes sobre el total de eventos
- `percentage_last_month_checkouts`: porcentaje de la cantidad de checkouts de ese mes sobre el total de eventos
- `percentage_last_month_viewed_products`: porcentaje de la cantidad de productos vistos de ese mes sobre el total de eventos

3.3.10. Días entre el último checkout y última actividad

La intención de este feature es medir la diferencia de días que tiene cada usuario entre la compra de un celular y la ultima vez que visualizo el producto comprado. De esta forma poder predecir en base a los productos vistos si es posbile que se haga una compra.

3.3.11. Estados de celulares

Utilizando la lógica de que hay empresas que compran celulares en mal estado con el único fin de usar sus partes como respuestos se plantea agregar una columna que indique porcentaje de celulares en estado Bom - Sem Touch ID vs Bom sobre todos los celulares vistos.

3.3.12. Varianza logarítmica de productos vistos

Se propone analizar la varianza en los precios de los productos visitados. Es decir, si los usuarios ven telefonos de un rango pequeño de precio o, por el contrario, articulos de precios muy variados. Se utilizó una escala logarítmica para seguir manteniendo las proporciones sin tener una gran diferencia entre la varianza de un usuario y la de otro.

3.3.13. ¿El usuario compró más de la media?

Se propone como feature evaluar si el usuario compró un celular por encima de la media de precios.

3.3.14. ¿Cuántas veces vio el último modelo que compró?

La idea de este atributo es evaluar una cierta correlación entre los usuarios de la cantidad de veces que se ve un modelo antes de comprarlo. Si un usuario ve una cantidad significativamente grande de veces un modelo es muy probable que lo compre. Esto no quiere decir que sea imposible que un usuario pueda ver una vez un modelo y no comprarlo o verlo muchas veces y no comprarlo pero se busca analizar el caso más general.

3.3.15. ¿Cuántas veces vio la última marca que compró?

Una lógica similar a la sección precedente es la que se sigue con este feature. La idea sería también analizar si un usuario se restringe a un modelo en particular o si puede ver distintos celulares de la misma marca y elegir uno de ellos.

3.3.16. Comportamiento en sesiones de las últimas semanas

Se presenta una idea similar a la de las secciones `3.9` y `3.10`:

3.3.17. ¿Cuántas veces entró cada usuario a las static pages?

3.3.18. ¿Cuántas veces vio el modelo que más vio la última semana?

3.4. Feature selection

Una vez que se tienen todos los atributos en un mismo dataframe y luego de un par de pruebas se ve que no siempre hay que entrenar los modelso con la totalidad del dataframe. Viendo que para algunos algoritmos el orden y la selección de los features logrababa distintos resultados, se busca la forma de encontrar la combinación óptima de features y eliminar todo el ruido posible.

El modelo usado de referencia para esta sección es la del Random Forest, debido a la estabilidad de este para mostrar la importancia de cada feature, ya que los árboles que crea el algoritmo toman distintos subconjuntos de atributos tomados al azar y arrojan distintos resultados. De esta manera, con cientos o miles de árboles el algoritmo adopta una amplia capacidad predictora de cada atributo.

Se utiliza como métrica el area bajo la curva debido a que es la utilizada por la plataforma de Kaggle para evaluar la eficiencia de los distintos modelos utilizados.

Las distintas formas de selección utilizadas son:

- **Métodos greedy**
  - **Cumulative Importance**

    El método es el más intuitivo para la selección de features debido a su naturaleza *greedy*. La idea es primero correr el Random Forest sobre todo el dataframe, y sobre este resultado obtener la importancia de cada feature. Ahora, con los features ordenados según importancia se genera una lista de listas que agrega un feature a la vez. Por ejemplo siendo `a,b,c` los features ordenados, se obtienen las listas `a`, `a,b` y `abc`.

    Ahora, con la lista de listas de features segun importancia, se corre un Random Forest para cada subconjunto, en busqueda del *codo*, es decir, el subconjunto que mayor AUC tenga antes de decaer a los features ruidosos.

    La desventaja que presenta este método es que parte del supuesto de que la importancia de cada feature es la misma para la corrida del dataframe entero que para subconjuntos de esto, lo cual no es cierto. De todas formas, es uno de los métodos que mejores resultados dió.

- **Métodos de fuerza bruta**
  - **Forward Selection**

    Con este método se comienza con ningún atributo y en cada paso se agrega el atributo que genere mejor resultado. Se agregan atributos siempre y cuando los resultados mejoren. El algoritmo termina cuando el resultado no se puede mejorar o cuando ya se han agregado todos los atributos.
  - **Backward Elimination**

    Este método funciona a la inversa de Forward Selection. Se comienza con todos los atributos y se quita en cada iteración el atributo que menos aportaba. De esta manera el algoritmo termina cuando al quitar un atributo el resultado empeora o cuando ya no hay más atributos por quitar.
  - **Stepwise Selection**

    Este método es una combinación de ambos métodos descritos anteriormente. La idea es partir de una lista vacía como en Forward Selection, y agregar siempre el mejor atributo posible (como Forward Selection). Luego, una vez que se tiene el mejor atributo, se busca recursivamente con Backward Elimination si algun atributo sobra. Es una forma sabia de moverse a traves de los atributos; en cada paso se agrega el mejor posible y saca todos los que empeoraban esa nueva combinación.

El mayor problema de estos tres métodos es que, si bien son efectivos en los subconjuntos encontrados, son muy costosos en tiempo (como todo algoritmo de fuerza bruta). Pensar que se hacen n iteraciones (siendo n la cantidad de features) y para cada una de esas iteraciones se hacen n más (y ni pensar en Stepwise Selection, que hace aun n más). Pasados los 100 features ya esto pasa a ser increiblemente costoso. Es por esto que se le agrega una condición de corte, donde si para las x iteraciones no se encuentra una mejora simplemente se corte el proceso. Si bien esto plantea la posibilidad de caer en un máximo local, en tiempo termina siendo rendidor.

- **Métodos de otras heurísticas**

  Se agregan algunos criterios más, para aumentar los subconjuntos a utilizar.

  - **Full Dataframe** Como su nombre lo dice, simplemente probar el modelo sobre el dataframe entero.
  - **Selección a Mano** Para no confiar solamente en los métodos algorítmicos, se hacen selecciones de los features manualmente.
  - **Random Selection** También se hacen selecciones azarosas de n atributos, para así en cada corrida ver si se encuentra alguna buena combinación que sirva para la selección manual o de una idea de que atributos ayudan o no.
  - **Feature Intersection** Aprovechando que se tienen muchas formas distintas de seleccionar subconjuntos, también se utiliza como método el intersectar todas estas combinaciones, con la idea de que si un feature aparece en varios de los subconjuntos anteriores, pues ciertamente será bueno por si mismo.

4. Modelos

4.1. Algoritmos de clasificación utilizados

4.1.1. Árboles de decisión

4.1.2. Random forests

4.1.3. XGBoost

4.1.4. Logistic regression

4.1.5. KNN

4.1.6. Naïve-Bayes

4.1.7. LightGBM

4.1.8. Redes neuronales

4.1.9. Catboost

Si bien se encontraron resultados decentes para este modelo, se decidió no utilizarlo en las últimas corridas por lo mucho que tardaba cada ejecución.

4.1.10. Gradient boosting

4.2. Parameter tuning

Griiiiidseaaaaarch

4.3. Ensambles

5. Desarrollo

5.1. Submission framework

Se define un framework y una serie de funciones para armar las postulaciones de predicciones del trabajo práctico. Las mismas siguen los siguientes pasos:

1. Creación de la matriz `X` y el vector `y` para entrenar
2. Generación del split para obtener los sets de entrenamiento y de prueba
3. Ejecución del algoritmo de Machine Learning que devuelve un dataframe con person como índice y los *labels* como única columna.
4. Se obtienen las 3 medidas utilizadas como métrica para evaluar el rendimiento del algoritmo: precisión, auc y aucpr.
5. Se predicen las probabilidades
6. Se observa información relevante de la ejecución como la importancia de los features elegidos
7. Se guardan los resultados como csv para ser submiteados

5.2. Encontrando el mejor submit

Una vez corridos todos los notebooks y creados todos los dataframes necesarios, finalmente se busca el mejor submit. Esta postulación esta compuesta por los dos componentes que se buscaron todo el trabajo: el subconjunto de **features** y el **modelo** a utlizar. El procedimiento para encontrarlo fue uno sencillo por fuerza bruta; utilizando todo lo generado en el trabajo simplemente hay que quedarse con lo que mejor puntaje tenga. La elección del algoritmo para realizar el *submit* se hace en base a todos los algoritmos y a combinaciones duales de ellos.

Simplemente se definen dos listas que luego se cruzaran en busqueda de la mejor combinación de ellas. Por un lado, `posibilidades_algoritmos_y_ensambles` y por el otro `posibilidades_features`.

Como referencia de puntaje se decidió utilizar la métrica de Area Under Curve (AUC), ya que luego de distintas pruebas (`precision_recall`, `accuracy_score`, `f1_score`) se vió que esta es la más fiel a lo buscado.

Por un lado, se definen todos los algoritmos con sus mejores hiper-parametros seleccionados. Luego, se hace tanto el bagging sobre ellos como el ensamble por mayoría de votos sobre combinaciones de a pares de ellos. También, se hacen algunas combinaciones de tres algoritmos por mayoría de votos.

Por el otro lado, se definen todos los subconjuntos de features seleccionados por los diversos métodos.

Finalmente, se corren todas las combinaciones de estas dos listas, un proceso sumamente lento y costoso tanto en tiempo como memoria, pero eventualmente este proceso devolverá una lista ordenada según AUC de las combinaciones. De estas, se deciden las que se van a postular y se opta por entrenarlas con todo el dataframe, en vez de con un set de entrenamiento, ya que si bien para el trabajo localmente el entrenamiento es un subconjunto del dataframe, para fines de la predicción se puede tomar a todo el dataframe como el entrenamiento.

Es importante notar que no siempre hay que postular el primer resultado, si no que hay que analizar cuales fueron estos resultados y por que, para evitar ser víctimas del overfitting.

[TO-DO plotssssss]

6. Resultados obtenidos

7. Conclusiones

its a kind of magic