




Organización de Datos 75.06/95.58
Trabajo Práctico N° 2

GRUPO: 15 - Leo Dattioli

Apellido y Nombre	Padrón	Correo electrónico
Marinero Santiago	97969	santiagomarinero1@gmail.com
Opizzi Juan Cruz	99807	juanopizzi@gmail.com
Lasanta Tito	99162	titonoolvida@gmail.com
Lizarraga Marin Augusto Joaquín	99636	augusjoaliza@hotmail.com

Github: 

<https://github.com/Santiago39353535/Tp2-Datos>

Índice

1. Introducción	2
2. Limpieza	2
3. Feature engineering	2
3.1. Reutilización de información del TP1	2
3.2. Información relacionada a eventos	2
3.3. Información relacionada a eventos y su tiempo	3
3.4. Información pertinente al usuario	3
3.5. Información relacionada a features ineficientes	3
4. Machine learning	3
4.1. Preparación de datos	3
4.2. Algoritmos Utilizados	4
4.3. Búsqueda de hiper-parámetros	5
5. Conclusión	5

1. Introducción

El objetivo de este trabajo es generar un modelo predictivo basados en algoritmos de Machine Learning para decidir si un usuario de Trocafone(www.trocafone.com.ar) realice una conversion(Es decir, una compra de un producto). Para cumplir este objetivo, se utilizo la información provista por Trocafone, para la generación de distintos features. En esta información, se halla distintos datos sobre los usuarios, como los eventos que realizaron(búsqueda de productos, checkout o la conversion misma) y los productos relacionados con estos.

2. Limpieza

Los datos provistos no parecen haber tenido outliers a simple vista, a su vez el intentar filtrar filas de información en estos datos con características distintivas. Por ejemplo, individuos con mas de 5 compras en el sitio terminaba empeorando el puntaje.

3. Feature engineering

3.1. Reutilización de información del TP1

Del trabajo practico anterior pudimos utilizar en su mayoría el código implementado que traduce un poco lo que vendría a ser los miles de versiones de navegador a solamente su marca, lo mismo para con los modelos de los celulares mas comunes. Adicionalmente creemos que se nos facilito el encontrar los features mas pertinentes para con la predicción teniendo ya conocimiento de la naturaleza de los datos, por ejemplo priorizamos crear features en base a eventos que tiene una cantidad relevante de datos, search por ejemplo fue uno de los eventos mas utilizados con este fin.

3.2. Información relacionada a eventos

Los primeros features que se nos ocurrieron fueron el encontrar los datos mas comunes para cada usuario, como el modelo que mas aparece en sus eventos, su buscador, navegador, etc. También contamos la cantidad de eventos que llevo a cabo cada usuario, de estos features algunos confundían a los predictores, como por ejemplo la cantidad de veces que aparecía "lead" por persona, y otros se convirtieron en la base del algoritmo como podría ser para el caso de checkout".

3.3. Información relacionada a eventos y su tiempo

Para poder marcar sucesos en el tiempo optamos por anotar la cantidad de tiempo restantes hasta el primer día a predecir. Esto parecería habernos servido al comienzo pero a medida que avanzamos en el desarrollo del modelo pareció no ser completamente óptimo, dado que llegado a un punto los features anotados de este modo compartían mucha de su información. Con el método descrito anteriormente creamos features como: Cuando fue la última vez que realizo un evento en específico, cuanto tardo en general entre eventos, cual fue el primer/ultimo día en el que ingresaron a Trocafone. Otras observaciones con respecto a la fecha, fueron el día de la semana en el que realizaron la mayor cantidad de eventos, que tantos eventos realizaron en tal día de la semana en total.

3.4. Información pertinente al usuario

En este sector los primeros features que se nos ocurrieron fueron el encontrar los datos mas comunes para cada usuario, como el modelo que mas aparece en sus eventos, su buscador, navegador, etc. Adicionalmente también quisimos contemplar en nuestros features si a lo largo del uso del sitio el usuario cambio sus gustos, esto lo intentamos plasmar en 3 features marcando la condición de celular que buscaban al comienzo, a mitad y al final de su experiencia en el sitio. Además, se crearon features con la columna 'New vs Returning', creando features para la contar las veces que visita el sitio y el estado de cada usuario (comienzo y final) hasta el primero de junio. En etapas posteriores del proyecto, fueron filtrados por su poca importancia en los algoritmos utilizados.

3.5. Información relacionada a features ineficientes

Para aquellos features que disminuían nuestra capacidad de predicción usamos una lista para filtrar dichos features, quedándonos solo con los que fueron mejorando nuestra predicción.

4. Machine learning

4.1. Preparación de datos

Para poder realizar la predicción con machine learning una vez creados los datos encontramos dos problemas: primero, la conversión de los datos categóricos a numéricos. La solución a esto fue un algoritmo visto en clase

mean, encoding con smoothing, el ajuste de los hiper-parámetros agrego mas tiempo al proceso de ajuste de hiper-parámetros. El segundo problema fue el completado de los datos faltantes por usuario, por ejemplo no para todos los usuarios tenemos registrado que navegador utilizaban, tomamos 2 caminas para solucionar este problema, el primero para variables categóricas creamos una nueva clase que contenía a todos estos, para los casos numéricos sin embargo no pudimos tomar el mismo camino por lo que optamos por rellenar los huecos vacíos con una variable estadística, sea la moda, mediana o promedio, dependiendo de cual parecería resultar mejor.

4.2. Algoritmos Utilizados

En esta sección se mostraran que algoritmos de machine learning fueron probados para la realización del trabajo practico. Los algoritmos que se probaron son ensambles, dado que casi siempre son mejores que el resultado da cualquier algoritmo individualmente. Por ejemplo uno de los algoritmos utilizados es bagging, que lo que hace es aplicar el mismo clasificador n veces y promediar sus resultados para obtener el resultado final. Bagging se tiene que usar en conjunto con bootstrapping siempre.

- Decision Tree: Nuestro primer algoritmo utilizado, fue rápidamente superado por otros modelos, específicamente randomforest y gradien boost.
- Random Forest: No fue usado para la entrega final ya que el resultado de la predicción dio una precisión del 0.83025, el cual es baja en comparación a los demás que usamos, ademas de sufrir lentitud en comparación al resto.
- Adaptive Boosting: En vista de que gradient boosting fue el mejor modelo, se probó Adaptive boosting, pero resulto en un puntaje bajo de 0,83185, por lo que fue descartado.
- Bagging: También se probó un algoritmo de bagging, siendo el estimador base el Random Forest. Esto dio mejores resultados que el anterior con un puntaje igual a 0.84901.
- Gradient Boost: este es el que fue utilizado para la entrega del trabajo practico, dado que de entre todos los usados fue el que mejor resultado da con una precisión de 0.86939.

4.3. Búsqueda de hiper-parámetros

Para la búsqueda de los hiper-parámetros se uso el GridSearchCV de la librería sklearn. La grilla de parámetros probada fue:

- Max-features: auto. Es el numero de features a considerar cuando se hace el split.
- N-estimators: 10,20,25,30,40,50,100. Es la cantidad de etapas de boosting a realizar.
- Loss: ls, lad, huber, quantile.
- Subsample: 0.5,0.8,0.9,1.0. Es la fracción de muestras a utilizar para que el fitting de individual base learners.

Los mejores resultados fueron con los siguientes hiper-parámetros:

- Max-features: auto.
- Loss: ls.
- N-estimators: 30
- Subsample: 1.0

5. Conclusión

En este trabajo practico se probaron distintos algoritmos para machine learning y se buscaron distintos features para la predicción. De los cuales podemos sacar la conclusión de que los ensambles, en general, funcionan mejor. Y la importancia de elegir que features utilizar a la hora de predecir, ya que, por ejemplo al agregar ciertos agregar features baja la precisión, por eso hay que probar las distintas combinaciones para quedarnos con los mejores resultados posibles.