ChannelAttribution

Release 2.0.0

Davide Altomare, David Loris

July 08, 2020

Table of Contents

- *Index*
- *Module Index*
- *Search Page*

Markov Model for Online Multi-Channel Attribution Problem Advertisers use a variety of online marketing channels to reach consumers and they want to know the degree each channel contributes to their marketing success. This is called online multichannel attribution problem. In many cases, advertisers approach this problem through some simple heuristics methods that do not take into account any customer interactions and often tend to underestimate the importance of small channels in marketing contribution. This package provides a function that approaches the attribution problem in a probabilistic way. It uses a k-order Markov representation to identify structural correlations in the customer journey data. This would allow advertisers to give a more reliable assessment of the marketing contribution of each channel. The approach basically follows the one presented in Eva Anderl, Ingo Becker, Florian v. Wangenheim, Jan H. Schumann (2014). Differently from them, we solved the estimation process using stochastic simulations. In this way it is also possible to take into account conversion values and their variability in the computation of the channel importance. The package also contains a function that estimates three heuristic models (first-touch, last-touch and linear-touch approach) for the same problem.

`ChannelAttribution.`**`auto_markov_model`** (*Data*, *var_path*, *var_conv*, *var_null*, *var_value=None*, *max_order=10*, *roc_npt=100*, *plot=False*, *nsim_start=100000.0*, *max_step=None*, *out_more=False*, *sep='>'*, *ncore=1*, *nfold=10*, *seed=0*, *conv_par=0.05*, *rate_step_sim=1.5*, *verbose=True*)

Parameters

Data : *DataFrame*
customer journeys.

var_path: string
column of Data containing paths.

var_conv : *string*
column of Data containing total conversions for each path.

var_null : *string*
column of Data containing total paths that do not lead to conversion.

var_value : *string, optional, default None*
column of Data containing revenue for each path

max_order : *int, default 10*
maximum Markov Model order to be considered.

roc_npt: int, default 100
number of points to be used for the approximation of roc curve.

plot: bool, default True
if True, a plot with penalized auc with respect to order will be displayed.

nsim_start : *int, default 1e5*
minimum number of simulations to be used in computation.

max_step : *int, default None*
maximum number of steps for a single simulated path. if NULL, it is the maximum number of steps found into Data.

out_more : *bool, default False*
if True, transition probabilities between channels and removal effects will be returned.

sep : *string, default ">"*
separator between the channels.

ncore : *int, default 1*
number of threads to be used in computation.

nfold : *int, default 10*
how many repetitions to be used to verify if convergence has been reached at each iteration.

seed : *int, default 0*
random seed. Giving this parameter the same value over different runs guarantees that results will not vary.

conv_par : *double, default 0.05*
convergence parameter for the algorithm. The estimation process ends when the percentage of variation of the results over different repetions is less than convergence parameter.

rate_step_sim : *double, default 0*
number of simulations used at each iteration is equal to the number of simulations used at previous iteration multiplied by rate_step_sim.

verbose : *bool, default True*
if True, additional information about process convergence will be shown.

Returns

list of DataFrames

result: Dataframe
(column) channel_name : channel names (column) total_conversions : conversions attributed to each channel (column) total_conversion_value : revenues attributed to each channel

transition_matrix : *DataFrame*
(column) channel_from: channel from (column) channel_to : channel to (column) transition_probability : transition probability from channel_from to channel_to

removal_effects:
(column) channel_name : channel names (column) removal_effects_conversion : removal effects for each channel calculated using total conversions (column) removal_effects_conversion_value : removal effects for each channel calculated using revenues

Examples
Load Data

```
>>> import pandas as pd
```

```
>>> from ChannelAttribution import *
>>> Data = pd.read_csv('https://raw.githubusercontent.com/DavideAltomare/\
>>> ChannelAttribution/master/ChannelAttribution/\
>>> src/cypack/data/Data.csv',sep=";")
```

Estimate an automatic Makov model

```
>>> auto_markov_model(Data, "path", "total_conversions", "total_null")
```

ChannelAttribution.**choose_order** (*Data, var_path, var_conv, var_null, max_order=10, sep='>', ncore=1, roc_npt=100, plot=True*)

Find the minimum Markov Model order that gives a good representation of customers' behaviour for data considered. It requires paths that do not lead to conversion as input. Minimum order is found maximizing a penalized area under ROC curve.

Parameters

Data : *DataFrame*
customer journeys.

var_path: string
column of Data containing paths.

var_conv : *string*
column of Data containing total conversions for each path.

var_null : *string*
column of Data containing total paths that do not lead to conversion.

max_order : *int, default 10*
maximum Markov Model order to be considered.

sep : *string, default ">"*
separator between the channels.

ncore : *int, default 1*
number of threads to be used in computation.

roc_npt: int, default 100
number of points to be used for the approximation of roc curve.

plot: bool, default True
if True, a plot with penalized auc with respect to order will be displayed.

Returns

list

roc : *list DataFrame one for each order considered*
(column) tpr: true positive rate (column) fpr: false positive rate

auc : *DataFrame with the following columns*
(column) order: markov model order (column) auc: area under the curve (column) pauc: penalized auc

suggested order : *int*
estimated best order

Examples

Estimate best makov model order for your data

Load Data

```
>>> import pandas as pd
>>> from ChannelAttribution import *
>>> Data = pd.read_csv('https://raw.githubusercontent.com/DavideAltomare/\
```

```
>>> ChannelAttribution/master/ChannelAttribution/\
>>> src/cypack/data/Data.csv',sep=";")
```

```
>>> choose_order(Data, var_path="path", var_conv="total_conversions",
var_null="total_null")
```

ChannelAttribution.**heuristic_models** (*Data, var_path, var_conv, var_value=None, sep='>'*)

Estimate three heuristic models (first-touch, last-touch and linear) from customer journey data.

Parameters

Data : *DataFrame*
customer journeys.

var_path: string
column of Data containing paths.

var_conv : *string*
column of Data containing total conversions for each path.

var_value : *string, optional, default None*
column of Data containing revenue for each path.

sep : *string, default ">"*
separator between the channels.

Returns

DataFrame
(column) channel_name : channel names (column) first_touch_conversions : conversions attributed to each channel using first touch attribution. (column) first_touch_value : revenues attributed to each channel using first touch attribution. (column) last_touch_conversions : conversions attributed to each channel using last touch attribution. (column) last_touch_value : revenues attributed to each channel using last touch attribution. (column) linear_touch_conversions : conversions attributed to each channel using linear attribution. (column) linear_touch_value : revenues attributed to each channel using linear attribution.

Examples

Load Data

```
>>> import pandas as pd
>>> from ChannelAttribution import *
>>> Data = pd.read_csv('https://raw.githubusercontent.com/DavideAltomare/\
>>> ChannelAttribution/master/ChannelAttribution/\
>>> src/cypack/data/Data.csv',sep=";")
```

Estimate heuristic models on total conversions

```
>>> heuristic_models(Data,"path","total_conversions")
```

Estimate heuristic models on total conversions and total revenues

```
>>> heuristic_models(Data,"path","total_conversions",\
>>> var_value="total_conversion_value")
```

ChannelAttribution.**markov_model** (*Data, var_path, var_conv, var_value=None, var_null=None, order=1, nsim_start=100000.0, max_step=None, out_more=False, sep='>', ncore=1, nfold=10, seed=0, conv_par=0.05, rate_step_sim=1.5, verbose=True*)

Estimate a k-order Markov model from customer journey data. Differently from markov_model, this function iterates estimation until a desidered convergence is reached and enables multiprocessing.

Parameters

Data : *DataFrame*
customer journeys.

var_path: string
column of Data containing paths.

var_conv : *string*
column of Data containing total conversions for each path.

var_value : *string, optional, default None*
column of Data containing revenue for each path.

var_null : *string*
column of Data containing total paths that do not lead to conversion.

order : *int, default 1*
Markov model order.

nsim_start : *int, default 1e5*
minimum number of simulations to be used in computation.

max_step : *int, default None*
maximum number of steps for a single simulated path. if NULL, it is the maximum number of steps found into Data.

out_more : *bool, default False*
if True, transition probabilities between channels and removal effects will be returned.

sep : *string, default ">"*
separator between the channels.

ncore : *int, default 1*
number of threads to be used in computation.

nfold : *int, default 10*
how many repetitions to be used to verify if convergence has been reached at each iteration.

seed : *int, default 0*
random seed. Giving this parameter the same value over different runs guarantees that results will not vary.

conv_par : *double, default 0.05*
convergence parameter for the algorithm. The estimation process ends when the percentage of variation of the results over different repetions is less than convergence parameter.

rate_step_sim : *double, default 0*
number of simulations used at each iteration is equal to the number of simulations used at previous iteration multiplied by rate_step_sim.

verbose : *bool, default True*
if True, additional information about process convergence will be shown.

Returns

list of DataFrames

result: Dataframe
(column) channel_name : channel names (column) total_conversions : conversions attributed to each channel (column) total_conversion_value : revenues attributed to each channel

transition_matrix : *DataFrame*

(column) channel_from: channel from (column) channel_to : channel to (column) transition_probability : transition probability from channel_from to channel_to

removal_effects:

(column) channel_name : channel names (column) removal_effects_conversion : removal effects for each channel calculated using total conversions (column) removal_effects_conversion_value : removal effects for each channel calculated using revenues

Examples

Load Data

```
>>> import pandas as pd
>>> from ChannelAttribution import *
>>> Data = pd.read_csv('https://raw.githubusercontent.com/DavideAltomare/\
>>> ChannelAttribution/master/ChannelAttribution/\
>>> src/cypack/data/Data.csv',sep=";")
```

Estimate a Makov model using total conversions

```
>>> markov_model(Data, "path", "total_conversions")
```

Estimate a Makov model using total conversions and revenues

```
>>> markov_model(Data, "path", "total_conversions",
var_value="total_conversion_value")
```

Estimate a Makov model using total conversions, revenues and paths that do not lead to conversions

```
>>> markov_model(Data, "path", "total_conversions",
var_value="total_conversion_value", var_null="total_null")
```

Estimate a Makov model returning transition matrix and removal effects

```
>>> markov_model(Data, "path", "total_conversions",
var_value="total_conversion_value", var_null="total_null", out_more=True)
```

Estimate a Markov model using 4 threads

```
>>> markov_model(Data, "path", "total_conversions",
var_value="total_conversion_value", ncore=4)
```

ChannelAttribution.**transition_matrix** (*Data, var_path, var_conv, var_null, order=1, sep='>', flg_equal=True*)

Estimate a k-order transition matrix from customer journey data.

Parameters

Data : *DataFrame*

customer journeys.

var_path: string

column of Data containing paths.

var_conv : *string*

column of Data containing total conversions for each path.

var_null : *string*

column of Data containing total paths that do not lead to conversion.

order : *int, default 1*
Markov model order.

sep : *string, default ">"*
separator between the channels.

flg_equal: bool, default True
if True, transitions from a channel to itself will be considered.

Returns

list of DataFrames

channels: Dataframe
(column) id_channel : channel ids (column) channel_name : channel names

transition_matrix : *DataFrame*
(column) channel_from: id channel from (column) channel_to : id channel to (column) transition_probability : transition probability from channel_from to channel_to

Examples
Load Data

```
>>> import pandas as pd
>>> from ChannelAttribution import *
>>> Data = pd.read_csv('https://raw.githubusercontent.com/DavideAltomare/\
>>> ChannelAttribution/master/ChannelAttribution/\
>>> src/cypack/data/Data.csv',sep=";")
```

Estimate a second-order transition matrix using total conversions and paths that do not lead to conversion

```
>>> transition_matrix(Data, "path", "total_conversions", var_null="total_null",
order=2)
```

C

A

C

H

M

T