Расчет ROI для разных моделей атрибуции

Давайте разберем детально, как рассчитывать наши атрибуционные модели с помощью Python.

Чтобы упростить задачу, помимо файла с данными скачайте файл attribution_modeling.ipynb, в нем находится примеры кода, который мы разберем ниже.

Код написан под 3 версию Python и полагается на 2 сторонние библитеки – pandas и matplitlib.
Если у вас их нет, установите их прямо из ноутбука, запустив такой код:

```
In [ ]:
1 !pip3 install pandas matplotlib
```

Также в начале файла дан разбор смысла формулы для time-decay модели. Ознакомьтесь с ним, чтобы лучше понимать дальнейшие расчеты.

Для выполнения задания, вам нужно ответить на все вопросы, которые даны в конце этого документа.

1. Сделайте импорт необходимых библиотек и посмотрите на исходные данные (также для удобства данные немного преобразованы):

```
In [1]:
1 import pandas as pd
2 from datetime import datetime, timedelta
3 import matplotlib.pyplot as plt
```

```
In [2]:
1  session_data = pd.read_csv('data.csv', delimiter=';')
2
3  # change data formats
4  session_data['date'] = [datetime.strptime(x, '%Y-%m-%d') for x in session_data['date']]
5  session_data[['cost', 'value']] = session_data[['cost', 'value']].astype(float)
6
7  # additional groupping
8  session_data = session_data.groupby(['userId', 'date', 'trafficSource'])['cost', 'value'].sum().reset_index()
9  session_data = session_data.sort_values(by=['userId', 'date'])
10 session_data.head()
```

Out[2]:

	userId	date	trafficSource	cost	value
0	user_10	2020-01-05	telegram / posts	15.75	215.0
1	user_10000	2020-01-17	yandex / cpc	8.50	0.0
2	user_10000	2020-01-19	google / cpc	8.25	0.0
3	user_1002	2020-01-03	telegram / posts	15.75	0.0
4	user_1003	2020-01-08	google / cpc	8.25	0.0

2. Так как time-decay модель учитывает время от даты касания, до даты транзакции, для удобства расчетов нам понадобится столбец с датой покупки, сделаем для этого вспомогательную таблицу только с пользователями и датами их покупок. При этом переименуем столбец date на столбец purchaseDate:

```
In [3]:
# таблица только с датами покупок, нужна для time-decay модели

purchases_only = session_data[session_data['value'] > 0][['userId', 'date']]
purchases_only = purchases_only.groupby('userId')['date'].max().reset_index()
purchases_only.columns = ['userId', 'purchaseDate']
```

3. Чтобы посчитать атрибуцию по линейной модели, нам нужно суммарную выручку от пользователя разделить по всем визитам в равных долях, для этого посчитаем общее количество визитов (totalSessions) и общую выручку с пользователя (totalValue). Для атрибуции по первому касанию, нам нужен столбец с порядковым номером визита. Эти расчеты, а также объединение исходных данных с датами покупок из предыдущего пункта представлены ниже:

```
In [4]:
# расчет вспомогательных колонок

session_data['totalSessions'] = session_data.groupby('userId')['date'].transform(lambda x: x.count())
session_data['totalValue'] = session_data.groupby('userId')['value'].transform(lambda x: x.sum())
session_data['sessionNumber'] = session_data.groupby('userId').cumcount() + 1

session_data = session_data.merge(purchases_only, on='userId', how='left')
```

4. Теперь посчитаем веса для каждого визита в зависимости от количества дней до момента покупки, чтобы потом использовать эти веса в time-decay модели:

```
In [5]:
# расчет весов time-decay модели

session_data['daysToPurchase'] = [(x - y).days if x else 0
                                  for x, y in zip(session_data['purchaseDate'], session_data['date'])]

session_data['timeDecayWeight'] = [2**(-x / y)
                                   for x, y in zip(session_data['daysToPurchase'], session_data['totalSessions'])]

session_data['timeDecayWeight'] = session_data['timeDecayWeight'] / session_data.groupby('userId')['timeDecayWeight
```

5. И наконец посчитаем ценность каждого визита пользователя согласно разным моделям атрибуции

```
In [7]:
# расчет выручки на основе различных моделей атрибуции

session_data['lastTouchValue'] = session_data['value']
session_data['firstTouchValue'] = [x if y == 1 else 0
                                   for x, y in zip(session_data['totalValue'], session_data['sessionNumber'])]
session_data['linearValue'] = session_data['totalValue'] / session_data['totalSessions']
session_data['timeDecayValue'] = session_data['totalValue'] * session_data['timeDecayWeight']

```

мы видим, что атрибуция по последнему касанию – это просто наша ценность, которая получена в варианте по умолчанию, это то, как считают в первую очередь.

Ценность по первому касанию смотрит на порядковый номер визита и назначает визиту с номером 1 всю выручку от пользователя. Линейная модель просто делит всю выручку пользователя на количество визитов и каждому визиту присваивает это значение.

А модель time-decay общую ценность пользователя распределяет на каждый визит пропорционально посчитанному ранее весу. Выше есть объяснение, как работают веса этой модели, если вы еще этого не сделали, то ознакомьтесь с ним.

6. Проверим наши расчеты:

In [8]:
```
# проверка корректности расчетов

print(session_data[['lastTouchValue', 'firstTouchValue', 'linearValue', 'timeDecayValue']].sum())
```

```
lastTouchValue     131345.0
firstTouchValue    131345.0
linearValue        131345.0
timeDecayValue     131345.0
dtype: float64
```

Суммарная выручка должна быть одинакова по всем моделям, мы ведь не можем взять какую-то выручку извне или куда-то потерять часть средств.

7. В результате мы можем посчитать ROI для каждого из способов атрибуции:

```
# финальная таблица

totals = session_data.groupby('trafficSource')[['cost', 'lastTouchValue', 'firstTouchValue', 'linearValue',
                                                'timeDecayValue']].sum()

totals['lastTouchROI'] = 100*(round(totals['lastTouchValue'] / totals['cost'], 4))
totals['firstTouchROI'] = 100*(round(totals['firstTouchValue'] / totals['cost'], 4))
totals['linearROI'] = 100*(round(totals['linearValue'] / totals['cost'], 4))
totals['timeDecayROI'] = 100*(round(totals['timeDecayValue'] / totals['cost'], 4))

totals[['cost', 'lastTouchROI', 'firstTouchROI', 'linearROI', 'timeDecayROI']]
```

trafficSource	cost	lastTouchROI	firstTouchROI	linearROI	timeDecayROI
facebook / video	28208.00	119.59	136.65	123.62	121.08
google / cpc	25137.75	169.30	163.54	169.77	178.20
telegram / posts	35374.50	82.91	71.97	78.05	78.77
vk / display	4498.75	96.58	103.42	91.51	78.79
yandex / cpc	14025.00	152.42	153.83	157.36	149.64

Ответьте на вопросы:

- как зависит ценность сессии от количества сессий у пользователя в линейной модели атрибуции?
- какая сессия для нас ценнее, согласно модели time-decay, которая произошла сутки назад или которая произошла две недели назад?
- в каком случае ценность на одну сессию снижается сильнее, когда у пользователя 5 сессий или когда у него 10 сессий, для модели time-decay?