# Анализ убытков приложения приложения Procrastinate Pro+.

Несмотря на огромные вложения в рекламу, последние несколько месяцев компания терпит убытки. Есть данные о пользователях, привлечённых с 1 мая по 27 октября 2019 года (посещениях сайта, данные о покупках, данные о затратах на рекламу).

Цель: разобраться в причинах и помочь компании выйти в плюс.

Задача изучить:

* откуда приходят пользователи и какими устройствами они пользуются,
* сколько стоит привлечение пользователей из различных рекламных каналов;
* сколько денег приносит каждый клиент,
* когда расходы на привлечение клиента окупаются,
* какие факторы мешают привлечению клиентов.

# Описание данных

В моем распоряжении три датасета. Файл visits\_info\_short.csv хранит лог сервера с информацией о посещениях сайта, orders\_info\_short.csv — информацию о заказах, а costs\_info\_short.csv — информацию о расходах на рекламу.

Структура visits\_info\_short.csv:

* User Id — уникальный идентификатор пользователя,
* Region — страна пользователя,
* Device — тип устройства пользователя,
* Channel — идентификатор источника перехода,
* Session Start — дата и время начала сессии,
* Session End — дата и время окончания сессии.

Структура orders\_info\_short.csv:

* User Id — уникальный идентификатор пользователя,
* Event Dt — дата и время покупки,
* Revenue — сумма заказа.

Структура costs\_info\_short.csv:

* dt — дата проведения рекламной кампании,
* Channel — идентификатор рекламного источника,
* costs — расходы на эту кампанию.

# План работ по проекту

* загрузка данных и подготовка их к анализу,
* написание функции для расчёта и анализа LTV, ROI, удержания и конверсии,
* исследовательский анализ данных,
* оценка окупаемости рекламы,
* выводы и рекоммендации.

Ввод [1]:

**import** pandas **as** pd

**import** numpy **as** np

**from** datetime **import** datetime, timedelta

**from** matplotlib **import** pyplot **as** plt

**import** warnings

warnings.filterwarnings('ignore') *#убираем сообщения о предупреждениях*

Ввод [2]:

visits **=** pd.read\_csv('/datasets/visits\_info\_short.csv') *# журнал сессий*

orders **=** pd.read\_csv('/datasets/orders\_info\_short.csv') *# заказы*

costs **=** pd.read\_csv('/datasets/costs\_info\_short.csv') *# траты на рекламу*

​

Ввод [3]:

visits.head()

Out[3]:

|  | **User Id** | **Region** | **Device** | **Channel** | **Session Start** | **Session End** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 981449118918 | United States | iPhone | organic | 2019-05-01 02:36:01 | 2019-05-01 02:45:01 |
| **1** | 278965908054 | United States | iPhone | organic | 2019-05-01 04:46:31 | 2019-05-01 04:47:35 |
| **2** | 590706206550 | United States | Mac | organic | 2019-05-01 14:09:25 | 2019-05-01 15:32:08 |
| **3** | 326433527971 | United States | Android | TipTop | 2019-05-01 00:29:59 | 2019-05-01 00:54:25 |
| **4** | 349773784594 | United States | Mac | organic | 2019-05-01 03:33:35 | 2019-05-01 03:57:40 |

Ввод [4]:

visits.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 309901 entries, 0 to 309900

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 User Id 309901 non-null int64

1 Region 309901 non-null object

2 Device 309901 non-null object

3 Channel 309901 non-null object

4 Session Start 309901 non-null object

5 Session End 309901 non-null object

dtypes: int64(1), object(5)

memory usage: 14.2+ MB

Ввод [5]:

visits.isna().sum()

Out[5]:

User Id 0

Region 0

Device 0

Channel 0

Session Start 0

Session End 0

dtype: int64

Ввод [6]:

visits.duplicated().sum()

Out[6]:

0

Во фрейме visits вижу некорректный тип данных в столбцах Session Start и Session End. Неоднородный регистр названий столбцов. Пропуски и дубликаты отсутствуют.

Ввод [7]:

orders.head()

Out[7]:

|  | **User Id** | **Event Dt** | **Revenue** |
| --- | --- | --- | --- |
| **0** | 188246423999 | 2019-05-01 23:09:52 | 4.99 |
| **1** | 174361394180 | 2019-05-01 12:24:04 | 4.99 |
| **2** | 529610067795 | 2019-05-01 11:34:04 | 4.99 |
| **3** | 319939546352 | 2019-05-01 15:34:40 | 4.99 |
| **4** | 366000285810 | 2019-05-01 13:59:51 | 4.99 |

Ввод [8]:

orders.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 40212 entries, 0 to 40211

Data columns (total 3 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 User Id 40212 non-null int64

1 Event Dt 40212 non-null object

2 Revenue 40212 non-null float64

dtypes: float64(1), int64(1), object(1)

memory usage: 942.6+ KB

Ввод [9]:

orders.isna().sum()

Out[9]:

User Id 0

Event Dt 0

Revenue 0

dtype: int64

Ввод [10]:

orders.duplicated().sum()

Out[10]:

0

Во фрейме orders вижу некорректный тип данных в столбце Event Dt. Неоднородный регистр названий столбцов. Пропуски и дубликаты отсутствуют.

Ввод [11]:

costs.head()

Out[11]:

|  | **dt** | **Channel** | **costs** |
| --- | --- | --- | --- |
| **0** | 2019-05-01 | FaceBoom | 113.3 |
| **1** | 2019-05-02 | FaceBoom | 78.1 |
| **2** | 2019-05-03 | FaceBoom | 85.8 |
| **3** | 2019-05-04 | FaceBoom | 136.4 |
| **4** | 2019-05-05 | FaceBoom | 122.1 |

Ввод [12]:

costs.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1800 entries, 0 to 1799

Data columns (total 3 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 dt 1800 non-null object

1 Channel 1800 non-null object

2 costs 1800 non-null float64

dtypes: float64(1), object(2)

memory usage: 42.3+ KB

Ввод [13]:

costs.isna().sum()

Out[13]:

dt 0

Channel 0

costs 0

dtype: int64

Ввод [14]:

costs.duplicated().sum()

Out[14]:

0

Во фрейме costs вижу некорректный тип данных в столбце dt. Неоднородный регистр названий столбцов. Пропуски и дубликаты отсутствуют.

Ввод [15]:

*#привожу названия столбцов к нижнему регистру*

visits.columns **=** visits.columns.str.lower()

orders.columns **=** orders.columns.str.lower()

costs.columns **=** costs.columns.str.lower()

​

costs.head()

Out[15]:

|  | **dt** | **channel** | **costs** |
| --- | --- | --- | --- |
| **0** | 2019-05-01 | FaceBoom | 113.3 |
| **1** | 2019-05-02 | FaceBoom | 78.1 |
| **2** | 2019-05-03 | FaceBoom | 85.8 |
| **3** | 2019-05-04 | FaceBoom | 136.4 |
| **4** | 2019-05-05 | FaceBoom | 122.1 |

Ввод [16]:

*#меняю названия столбцов*

visits.columns **=** ['user\_id', 'region', 'device', 'channel', 'session\_start', 'session\_end']

orders.columns **=** ['user\_id', 'event\_dt', 'revenue']

orders.head()

Out[16]:

|  | **user\_id** | **event\_dt** | **revenue** |
| --- | --- | --- | --- |
| **0** | 188246423999 | 2019-05-01 23:09:52 | 4.99 |
| **1** | 174361394180 | 2019-05-01 12:24:04 | 4.99 |
| **2** | 529610067795 | 2019-05-01 11:34:04 | 4.99 |
| **3** | 319939546352 | 2019-05-01 15:34:40 | 4.99 |
| **4** | 366000285810 | 2019-05-01 13:59:51 | 4.99 |

Ввод [17]:

*# преобразую данные о времени*

visits['session\_start'] **=** pd.to\_datetime(visits['session\_start'])

visits['session\_end'] **=** pd.to\_datetime(visits['session\_end'])

​

orders['event\_dt'] **=** pd.to\_datetime(orders['event\_dt'])

costs['dt'] **=** pd.to\_datetime(costs['dt']).dt.date

​

visits.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 309901 entries, 0 to 309900

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 user\_id 309901 non-null int64

1 region 309901 non-null object

2 device 309901 non-null object

3 channel 309901 non-null object

4 session\_start 309901 non-null datetime64[ns]

5 session\_end 309901 non-null datetime64[ns]

dtypes: datetime64[ns](2), int64(1), object(3)

memory usage: 14.2+ MB

В данных отсутствуют пропуски и дубликаты; в ходе предобработки названия столбцов были приведены к стандартному стилю, тип данных изменен там где это необходимо для дальнейших рассчетов.

### 0.1  Задайте функции для расчёта и анализа LTV, ROI, удержания и конверсии.

Разрешается использовать функции, с которыми вы познакомились в теоретических уроках.

Это функции для вычисления значений метрик:

* get\_profiles() — для создания профилей пользователей,
* get\_retention() — для подсчёта Retention Rate,
* get\_conversion() — для подсчёта конверсии,
* get\_ltv() — для подсчёта LTV.

А также функции для построения графиков:

* filter\_data() — для сглаживания данных,
* plot\_retention() — для построения графика Retention Rate,
* plot\_conversion() — для построения графика конверсии,
* plot\_ltv\_roi — для визуализации LTV и ROI.

# Функции для расчета и анализа LTV, ROI, удержания и конверсии

## 1  Функция для расчёта LTV и ROI

Ввод [18]:

**def** get\_ltv(

profiles,

purchases,

observation\_date,

horizon\_days,

dimensions**=**[],

ignore\_horizon**=False**,

):

​

*# исключаем пользователей, не «доживших» до горизонта анализа*

last\_suitable\_acquisition\_date **=** observation\_date

**if** **not** ignore\_horizon:

last\_suitable\_acquisition\_date **=** observation\_date **-** timedelta(

days**=**horizon\_days **-** 1

)

result\_raw **=** profiles.query('dt <= @last\_suitable\_acquisition\_date')

*# добавляем данные о покупках в профили*

result\_raw **=** result\_raw.merge(

purchases[['user\_id', 'event\_dt', 'revenue']], on**=**'user\_id', how**=**'left'

)

*# рассчитываем лайфтайм пользователя для каждой покупки*

result\_raw['lifetime'] **=** (

result\_raw['event\_dt'] **-** result\_raw['first\_ts']

).dt.days

*# группируем по cohort, если в dimensions ничего нет*

**if** len(dimensions) **==** 0:

result\_raw['cohort'] **=** 'All users'

dimensions **=** dimensions **+** ['cohort']

​

*# функция группировки по желаемым признакам*

**def** group\_by\_dimensions(df, dims, horizon\_days):

*# строим «треугольную» таблицу выручки*

result **=** df.pivot\_table(

index**=**dims, columns**=**'lifetime', values**=**'revenue', aggfunc**=**'sum'

)

*# находим сумму выручки с накоплением*

result **=** result.fillna(0).cumsum(axis**=**1)

*# вычисляем размеры когорт*

cohort\_sizes **=** (

df.groupby(dims)

.agg({'user\_id': 'nunique'})

.rename(columns**=**{'user\_id': 'cohort\_size'})

)

*# объединяем размеры когорт и таблицу выручки*

result **=** cohort\_sizes.merge(result, on**=**dims, how**=**'left').fillna(0)

*# считаем LTV: делим каждую «ячейку» в строке на размер когорты*

result **=** result.div(result['cohort\_size'], axis**=**0)

*# исключаем все лайфтаймы, превышающие горизонт анализа*

result **=** result[['cohort\_size'] **+** list(range(horizon\_days))]

*# восстанавливаем размеры когорт*

result['cohort\_size'] **=** cohort\_sizes

​

*# собираем датафрейм с данными пользователей и значениями CAC,*

*# добавляя параметры из dimensions*

cac **=** df[['user\_id', 'acquisition\_cost'] **+** dims].drop\_duplicates()

​

*# считаем средний CAC по параметрам из dimensions*

cac **=** (

cac.groupby(dims)

.agg({'acquisition\_cost': 'mean'})

.rename(columns**=**{'acquisition\_cost': 'cac'})

)

*# считаем ROI: делим LTV на CAC*

roi **=** result.div(cac['cac'], axis**=**0)

​

*# удаляем строки с бесконечным ROI*

roi **=** roi[**~**roi['cohort\_size'].isin([np.inf])]

​

*# восстанавливаем размеры когорт в таблице ROI*

roi['cohort\_size'] **=** cohort\_sizes

​

*# добавляем CAC в таблицу ROI*

roi['cac'] **=** cac['cac']

​

*# в финальной таблице оставляем размеры когорт, CAC*

*# и ROI в лайфтаймы, не превышающие горизонт анализа*

roi **=** roi[['cohort\_size', 'cac'] **+** list(range(horizon\_days))]

​

*# возвращаем таблицы LTV и ROI*

**return** result, roi

​

*# получаем таблицы LTV и ROI*

result\_grouped, roi\_grouped **=** group\_by\_dimensions(

result\_raw, dimensions, horizon\_days

)

​

*# для таблиц динамики убираем 'cohort' из dimensions*

**if** 'cohort' **in** dimensions:

dimensions **=** []

*# получаем таблицы динамики LTV и ROI*

result\_in\_time, roi\_in\_time **=** group\_by\_dimensions(

result\_raw, dimensions **+** ['dt'], horizon\_days

)

​

**return** (

result\_raw, *# сырые данные*

result\_grouped, *# таблица LTV*

result\_in\_time, *# таблица динамики LTV*

roi\_grouped, *# таблица ROI*

roi\_in\_time, *# таблица динамики ROI*

)

## 2  Функция для расчёта удержания

Ввод [19]:

**def** get\_retention(

profiles,

sessions,

observation\_date,

horizon\_days,

dimensions**=**[],

ignore\_horizon**=False**,

):

​

*# добавляем столбец payer в передаваемый dimensions список*

dimensions **=** ['payer'] **+** dimensions

​

*# исключаем пользователей, не «доживших» до горизонта анализа*

last\_suitable\_acquisition\_date **=** observation\_date

**if** **not** ignore\_horizon:

last\_suitable\_acquisition\_date **=** observation\_date **-** timedelta(

days**=**horizon\_days **-** 1

)

result\_raw **=** profiles.query('dt <= @last\_suitable\_acquisition\_date')

​

*# собираем «сырые» данные для расчёта удержания*

result\_raw **=** result\_raw.merge(

sessions[['user\_id', 'session\_start']], on**=**'user\_id', how**=**'left'

)

result\_raw['lifetime'] **=** (

result\_raw['session\_start'] **-** result\_raw['first\_ts']

).dt.days

*# функция для группировки таблицы по желаемым признакам*

**def** group\_by\_dimensions(df, dims, horizon\_days):

result **=** df.pivot\_table(

index**=**dims, columns**=**'lifetime', values**=**'user\_id', aggfunc**=**'nunique'

)

cohort\_sizes **=** (

df.groupby(dims)

.agg({'user\_id': 'nunique'})

.rename(columns**=**{'user\_id': 'cohort\_size'})

)

result **=** cohort\_sizes.merge(result, on**=**dims, how**=**'left').fillna(0)

result **=** result.div(result['cohort\_size'], axis**=**0)

result **=** result[['cohort\_size'] **+** list(range(horizon\_days))]

result['cohort\_size'] **=** cohort\_sizes

**return** result

​

*# получаем таблицу удержания*

result\_grouped **=** group\_by\_dimensions(result\_raw, dimensions, horizon\_days)

​

*# получаем таблицу динамики удержания*

result\_in\_time **=** group\_by\_dimensions(

result\_raw, dimensions **+** ['dt'], horizon\_days

)

​

*# возвращаем обе таблицы и сырые данные*

**return** result\_raw, result\_grouped, result\_in\_time

## 3  Функция для расчёта конверсии

Ввод [20]:

**def** get\_conversion(

profiles,

purchases,

observation\_date,

horizon\_days,

dimensions**=**[],

ignore\_horizon**=False**,

):

​

*# исключаем пользователей, не «доживших» до горизонта анализа*

last\_suitable\_acquisition\_date **=** observation\_date

**if** **not** ignore\_horizon:

last\_suitable\_acquisition\_date **=** observation\_date **-** timedelta(

days**=**horizon\_days **-** 1

)

result\_raw **=** profiles.query('dt <= @last\_suitable\_acquisition\_date')

​

*# определяем дату и время первой покупки для каждого пользователя*

first\_purchases **=** (

purchases.sort\_values(by**=**['user\_id', 'event\_dt'])

.groupby('user\_id')

.agg({'event\_dt': 'first'})

.reset\_index()

)

​

*# добавляем данные о покупках в профили*

result\_raw **=** result\_raw.merge(

first\_purchases[['user\_id', 'event\_dt']], on**=**'user\_id', how**=**'left'

)

​

*# рассчитываем лайфтайм для каждой покупки*

result\_raw['lifetime'] **=** (

result\_raw['event\_dt'] **-** result\_raw['first\_ts']

).dt.days

​

*# группируем по cohort, если в dimensions ничего нет*

**if** len(dimensions) **==** 0:

result\_raw['cohort'] **=** 'All users'

dimensions **=** dimensions **+** ['cohort']

​

*# функция для группировки таблицы по желаемым признакам*

**def** group\_by\_dimensions(df, dims, horizon\_days):

result **=** df.pivot\_table(

index**=**dims, columns**=**'lifetime', values**=**'user\_id', aggfunc**=**'nunique'

)

result **=** result.fillna(0).cumsum(axis**=**1)

cohort\_sizes **=** (

df.groupby(dims)

.agg({'user\_id': 'nunique'})

.rename(columns**=**{'user\_id': 'cohort\_size'})

)

result **=** cohort\_sizes.merge(result, on**=**dims, how**=**'left').fillna(0)

*# делим каждую «ячейку» в строке на размер когорты*

*# и получаем conversion rate*

result **=** result.div(result['cohort\_size'], axis**=**0)

result **=** result[['cohort\_size'] **+** list(range(horizon\_days))]

result['cohort\_size'] **=** cohort\_sizes

**return** result

*# получаем таблицу конверсии*

result\_grouped **=** group\_by\_dimensions(result\_raw, dimensions, horizon\_days)

​

*# для таблицы динамики конверсии убираем 'cohort' из dimensions*

**if** 'cohort' **in** dimensions:

dimensions **=** []

​

*# получаем таблицу динамики конверсии*

result\_in\_time **=** group\_by\_dimensions(

result\_raw, dimensions **+** ['dt'], horizon\_days

)

​

*# возвращаем обе таблицы и сырые данные*

**return** result\_raw, result\_grouped, result\_in\_time

Созданы функции для расчета и анализа LTV (Lifetime Value, пожизненная ценность), ROI (Return On Investment, окупаемость инвестиций), Retention Rate (коэффициент удержания) и Conversion Rate (коэффициент конверсии). LTV (Lifetime Value) — это пожизненная ценность клиента. LTV показывает прибыль от отношений с клиентом за весь период — с момента, когда он увидел первую рекламу или зарегистрировался на сайте, до последней покупки. ROI («Return On Investment», возврат инвестиций) — коэффициент рентабельности инвестиций, который помогает рассчитать окупаемость вложений в проект. Retention rate (RR) — коэффициент удержания клиентов. Показывает, насколько хорошо компания выстраивает долгосрочные отношения с клиентами. Коэффициент конверсии (conversion rate) — это доля пользователей, которые совершили целевое действие — зарегистрировались, подписались на рассылку, заполнили форму, оформили покупку — в общем количестве посетителей сайта.

### 3.1  Исследовательский анализ данных

* Составьте профили пользователей. Определите минимальную и максимальную даты привлечения пользователей.
* Выясните, из каких стран пользователи приходят в приложение и на какую страну приходится больше всего платящих пользователей. Постройте таблицу, отражающую количество пользователей и долю платящих из каждой страны.
* Узнайте, какими устройствами пользуются клиенты и какие устройства предпочитают платящие пользователи. Постройте таблицу, отражающую количество пользователей и долю платящих для каждого устройства.
* Изучите рекламные источники привлечения и определите каналы, из которых пришло больше всего платящих пользователей. Постройте таблицу, отражающую количество пользователей и долю платящих для каждого канала привлечения.

После каждого пункта сформулируйте выводы.

Ввод [21]:

*# момент анализа*

observation\_date **=** datetime(2019, 11, 1).date()

observation\_date

Out[21]:

datetime.date(2019, 11, 1)

Ввод [22]:

*# горизонт анализа*

horizon\_days **=** 14

Ввод [23]:

*# функция создания профилей*

**def** get\_profiles(visits, orders, costs):

​

*# находим параметры первых посещений*

profiles **=** (

visits.sort\_values(by**=**['user\_id', 'session\_start'])

.groupby('user\_id')

.agg(

{

'session\_start': 'first',

'channel': 'first',

'device': 'first',

'region': 'first',

}

)

.rename(columns**=**{'session\_start': 'first\_ts'})

.reset\_index()

)

​

*# для когортного анализа определяем дату первого посещения и первый день месяца, в который это посещение произошло*

profiles['dt'] **=** profiles['first\_ts'].dt.date

profiles['month'] **=** profiles['first\_ts'].astype('datetime64[M]')

​

*# добавляем признак платящих пользователей*

profiles['payer'] **=** profiles['user\_id'].isin(orders['user\_id'].unique())

​

*# считаем количество уникальных пользователей с одинаковыми источником и датой привлечения*

new\_users **=** (

profiles.groupby(['dt', 'channel'])

.agg({'user\_id': 'nunique'})

.rename(columns**=**{'user\_id': 'unique\_users'})

.reset\_index()

)

*# объединяем траты на рекламу и число привлечённых пользователей*

costs **=** costs.merge(new\_users, on**=**['dt', 'channel'], how**=**'left')

*#costs['acquisition\_cost'] = costs.set\_index('key').join(new\_users.set\_index('key'))*

​

*# делим рекламные расходы на число привлечённых пользователей*

costs['acquisition\_cost'] **=** costs['costs'] **/** costs['unique\_users']

​

*# добавляем стоимость привлечения в профили*

profiles **=** profiles.merge(

costs[['dt', 'channel', 'acquisition\_cost']],

on**=**['dt', 'channel'],

how**=**'left',

)

​

*# стоимость привлечения органических пользователей равна нулю*

profiles['acquisition\_cost'] **=** profiles['acquisition\_cost'].fillna(0)

​

**return** profiles

​

​

profiles **=** get\_profiles(visits, orders, costs)

profiles.head()

Out[23]:

|  | **user\_id** | **first\_ts** | **channel** | **device** | **region** | **dt** | **month** | **payer** | **acquisition\_cost** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 599326 | 2019-05-07 20:58:57 | FaceBoom | Mac | United States | 2019-05-07 | 2019-05-01 | True | 1.088172 |
| **1** | 4919697 | 2019-07-09 12:46:07 | FaceBoom | iPhone | United States | 2019-07-09 | 2019-07-01 | False | 1.107237 |
| **2** | 6085896 | 2019-10-01 09:58:33 | organic | iPhone | France | 2019-10-01 | 2019-10-01 | False | 0.000000 |
| **3** | 22593348 | 2019-08-22 21:35:48 | AdNonSense | PC | Germany | 2019-08-22 | 2019-08-01 | False | 0.988235 |
| **4** | 31989216 | 2019-10-02 00:07:44 | YRabbit | iPhone | United States | 2019-10-02 | 2019-10-01 | False | 0.230769 |

По условиям задачи мы смотрим данные 1 ноября 2019 года. В организации принято считать, что окупаемость должна наступать не позднее, чем через 2 недели после привлечения пользователей. Таким образом, момент анализа составляет 1 ноября 2019 г., горизонт анализа - 14 дней. Портрет пользователя состоит из таких данных как его уникальный номер, дача первого посещения, название канала по которому он пришел, тип устройства которым пользовался, регион, платежный статус, сумма потраченная на привлечение клиента.

Ввод [24]:

*#определяю минимальную и максимальную даты привлечения пользователей*

min\_analysis\_date **=** profiles['dt'].min()

max\_analysis\_date **=** profiles['dt'].max()

display(min\_analysis\_date, max\_analysis\_date)

datetime.date(2019, 5, 1)

datetime.date(2019, 10, 27)

Ввод [25]:

region **=** profiles.groupby('region').agg({'user\_id': 'nunique', 'payer': 'sum'}).sort\_values(by**=**'payer', ascending**=False**)

region['share'] **=** round(region.payer **/** region.user\_id, 3)

region

Out[25]:

|  | **user\_id** | **payer** | **share** |
| --- | --- | --- | --- |
| **region** |  |  |  |
| **United States** | 100002 | 6902 | 0.069 |
| **UK** | 17575 | 700 | 0.040 |
| **France** | 17450 | 663 | 0.038 |
| **Germany** | 14981 | 616 | 0.041 |

Пользователи приходят в приложение из таких стран как США, Германии, Великобритании и Франции. Больше всего платящих пользователей приходится на США.

Ввод [26]:

device **=** profiles.groupby('device').agg({'user\_id': 'nunique', 'payer': 'sum'}).sort\_values(by**=**'payer', ascending**=False**)

device['share'] **=** round(device.payer **/** device.user\_id, 2)

device

Out[26]:

|  | **user\_id** | **payer** | **share** |
| --- | --- | --- | --- |
| **device** |  |  |  |
| **iPhone** | 54479 | 3382 | 0.06 |
| **Android** | 35032 | 2050 | 0.06 |
| **Mac** | 30042 | 1912 | 0.06 |
| **PC** | 30455 | 1537 | 0.05 |

Пользователи предпочитают использовать такие устройства как iPhone, Android, Mac и PC - наибольшее количество платящих пользователей используют iPhone.

Ввод [27]:

channel **=** profiles.groupby('channel').agg({'user\_id': 'nunique', 'payer': 'sum'}).sort\_values(by**=**'payer', ascending**=False**)

channel['share'] **=** round(channel.payer **/** channel.user\_id, 2)

channel

Out[27]:

|  | **user\_id** | **payer** | **share** |
| --- | --- | --- | --- |
| **channel** |  |  |  |
| **FaceBoom** | 29144 | 3557 | 0.12 |
| **TipTop** | 19561 | 1878 | 0.10 |
| **organic** | 56439 | 1160 | 0.02 |
| **WahooNetBanner** | 8553 | 453 | 0.05 |
| **AdNonSense** | 3880 | 440 | 0.11 |
| **RocketSuperAds** | 4448 | 352 | 0.08 |
| **LeapBob** | 8553 | 262 | 0.03 |
| **OppleCreativeMedia** | 8605 | 233 | 0.03 |
| **lambdaMediaAds** | 2149 | 225 | 0.10 |
| **YRabbit** | 4312 | 165 | 0.04 |
| **MediaTornado** | 4364 | 156 | 0.04 |

Для привлечения клиентов используются такие рекламные источники как FaceBoom, TipTop, WahooNetBanne, AdNonSense, RocketSuperAds, LeapBob, OppleCreativeMedia, lambdaMediaAds, YRabbit, MediaTornado, из которых больше всего прибыли приносит FaceBoom, а наименее прибыльным является MediaTornado. Так же хочу заметить, что органические пользователи как источник прибыли входят в тройку лидеров по прибыльности.

### 3.2  Маркетинг

* Посчитайте общую сумму расходов на маркетинг.
* Выясните, как траты распределены по рекламным источникам, то есть сколько денег потратили на каждый источник.
* Постройте визуализацию динамики изменения расходов во времени (по неделям и месяцам) по каждому источнику. Постарайтесь отразить это на одном графике.
* Узнайте, сколько в среднем стоило привлечение одного пользователя (CAC) из каждого источника. Используйте профили пользователей.

Напишите промежуточные выводы.

Ввод [28]:

print('Всего в маркетинг вложено', costs['costs'].sum().round(1))

Всего в маркетинг вложено 105497.3

Ввод [29]:

*# потрачено денег на каждый источник*

costs.groupby('channel').agg({'costs': 'sum'}).sort\_values(by**=**'costs', ascending**=False**)

Out[29]:

|  | **costs** |
| --- | --- |
| **channel** |  |
| **TipTop** | 54751.30 |
| **FaceBoom** | 32445.60 |
| **WahooNetBanner** | 5151.00 |
| **AdNonSense** | 3911.25 |
| **OppleCreativeMedia** | 2151.25 |
| **RocketSuperAds** | 1833.00 |
| **LeapBob** | 1797.60 |
| **lambdaMediaAds** | 1557.60 |
| **MediaTornado** | 954.48 |
| **YRabbit** | 944.22 |

Всего 10 рекламных источников: больше всего вложено денег в рекламу на TipTop - 54751.30, меньше всего в рекламу на YRabbit - 944.22.

Ввод [30]:

*# создание новых столбцов и преобразование типа данных*

costs['dt'] **=** pd.to\_datetime(costs['dt'])

costs['month'] **=** costs['dt'].dt.month

costs['week'] **=** costs['dt'].dt.week

Ввод [31]:

costs.head()

Out[31]:

|  | **dt** | **channel** | **costs** | **month** | **week** |
| --- | --- | --- | --- | --- | --- |
| **0** | 2019-05-01 | FaceBoom | 113.3 | 5 | 18 |
| **1** | 2019-05-02 | FaceBoom | 78.1 | 5 | 18 |
| **2** | 2019-05-03 | FaceBoom | 85.8 | 5 | 18 |
| **3** | 2019-05-04 | FaceBoom | 136.4 | 5 | 18 |
| **4** | 2019-05-05 | FaceBoom | 122.1 | 5 | 18 |

Ввод [32]:

*# добавление категории Топ-2 и Топ-8 (название Топ в данном случае условно, имеются ввиду оставшиеся 8 каналов)*

top\_2 **=** ['TipTop', 'FaceBoom']

top\_8 **=** ['WahooNetBanner', 'AdNonSense', 'RocketSuperAds', 'LeapBob', 'OppleCreativeMedia', 'lambdaMediaAds']

​

*# группировка данных по неделям*

costs\_weekly **=** costs.groupby(['channel', pd.Grouper(key**=**'dt', freq**=**'W-MON')])['costs'].sum().reset\_index()

​

*# группировка данных по месяцам*

costs\_monthly **=** costs.groupby(['channel', pd.Grouper(key**=**'dt', freq**=**'M')])['costs'].sum().reset\_index()

​

*# построение графиков*

fig, axs **=** plt.subplots(nrows**=**2, ncols**=**2, figsize**=**(15, 10))

​

*# топ-2 по неделям и месяцам*

top\_2\_weekly **=** costs\_weekly[costs\_weekly['channel'].isin(top\_2)]

top\_2\_monthly **=** costs\_monthly[costs\_monthly['channel'].isin(top\_2)]

​

**for** ax, costs, title **in** zip(axs.flatten()[:2], [top\_2\_weekly, top\_2\_monthly], ['Top-2 Weekly Costs', 'Top-2 Monthly Costs']):

**for** channel **in** top\_2:

channel\_costs **=** costs[costs['channel'] **==** channel]

ax.plot(channel\_costs['dt'], channel\_costs['costs'], label**=**channel)

ax.set\_title(title)

ax.legend()

​

*# топ-8 по неделям и месяцам*

top\_8\_weekly **=** costs\_weekly[costs\_weekly['channel'].isin(top\_8)]

top\_8\_monthly **=** costs\_monthly[costs\_monthly['channel'].isin(top\_8)]

​

**for** ax, costs, title **in** zip(axs.flatten()[2:], [top\_8\_weekly, top\_8\_monthly], ['Top-8 Weekly Costs', 'Top-8 Monthly Costs']):

**for** channel **in** top\_8:

channel\_costs **=** costs[costs['channel'] **==** channel]

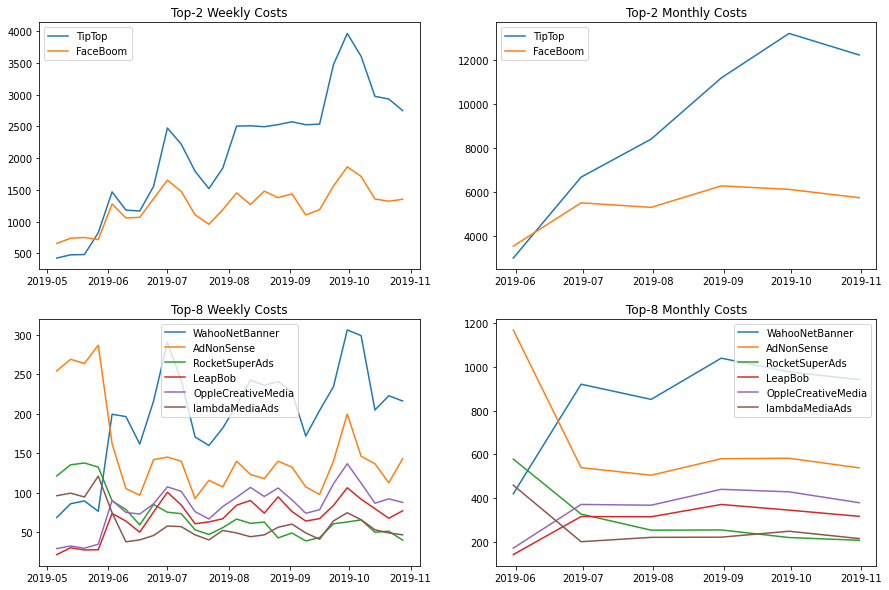
ax.plot(channel\_costs['dt'], channel\_costs['costs'], label**=**channel)

ax.set\_title(title)

ax.legend()

​

plt.show()



На графиках мы можем наблюдать довольно резкий рост расходов на маркетинг через канал TipTop с мая по сентябрь, меньше трат было на рекламу через FaceBoom, в начале анализируемого периода 3 место по расходом на рекламу делили между собой WahooNetBanne и AdNonSense, но в июле расходы на последний из двух источников значительно снизились и он опустился на 4 место; пользователей, пришедших по каналу organic мы не учитываем, тк расходы на их привлечение равны нулю. Пики изменений расходов на графиках по месяцам гораздо менее интенсивные, тк сглажены шагом в 30 дней.

Ввод [33]:

*# находим LTV с разбивкой по каналам привлечения*

dimensions **=** ['channel']

​

ltv\_raw, ltv\_grouped, ltv\_history, roi\_grouped, roi\_history **=** get\_ltv(

profiles, orders, observation\_date, horizon\_days, dimensions**=**dimensions

)

Ввод [34]:

*# находим максимальную дату привлечения из сырых данных LTV*

max\_acquitision\_dt **=** ltv\_raw['dt'].max()

*# отсекаем профили, которые «старше» этой даты*

ltv\_profiles **=** profiles.query('dt <= @max\_acquitision\_dt')

​

*# оставшееся число пользователей на каждый лайфтайм*

ltv\_profiles.groupby('dt').agg({'user\_id': 'nunique'})

Out[34]:

|  | **user\_id** |
| --- | --- |
| **dt** |  |
| **2019-05-01** | 909 |
| **2019-05-02** | 758 |
| **2019-05-03** | 849 |
| **2019-05-04** | 1015 |
| **2019-05-05** | 970 |
| **...** | ... |
| **2019-10-15** | 674 |
| **2019-10-16** | 656 |
| **2019-10-17** | 703 |
| **2019-10-18** | 911 |
| **2019-10-19** | 911 |

172 rows × 1 columns

Ввод [35]:

*# средний CAC по каналам привлечения*

cac **=** (

ltv\_profiles.groupby('channel')

.agg({'acquisition\_cost': 'mean'})

.rename(columns**=**{'acquisition\_cost': 'cac'}).sort\_values('cac', ascending**=False**)

)

round(cac, 2)

Out[35]:

|  | **cac** |
| --- | --- |
| **channel** |  |
| **TipTop** | 2.76 |
| **FaceBoom** | 1.11 |
| **AdNonSense** | 1.01 |
| **lambdaMediaAds** | 0.73 |
| **WahooNetBanner** | 0.60 |
| **RocketSuperAds** | 0.42 |
| **OppleCreativeMedia** | 0.25 |
| **MediaTornado** | 0.22 |
| **YRabbit** | 0.22 |
| **LeapBob** | 0.21 |
| **organic** | 0.00 |

Ввод [36]:

*# средний CAC по всему проекту без органических пользователей*

profiles\_noorganic **=** profiles[profiles['channel'] **!=** 'organic']

print('Средний CAC по всему проекту без органических пользователей:', profiles\_noorganic['acquisition\_cost'].mean())

Средний CAC по всему проекту без органических пользователей: 1.1274813239427588

Ввод [37]:

roi\_history.pivot\_table(

index**=**'dt', columns**=**'channel', values**=**'cac', aggfunc**=**'mean'

).plot(grid**=True**, figsize**=**(10, 5))

​

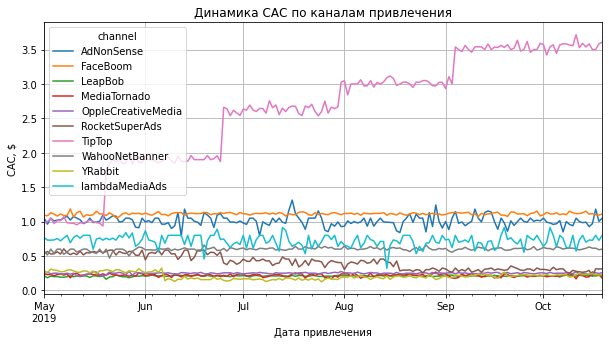
plt.ylabel('CAC, $')

plt.xlabel('Дата привлечения')

plt.title('Динамика САС по каналам привлечения')

plt.show()

​



Общая сумма расходов на рекламу составила 105497.3. Лидером по привлечению инвестиций стали каналы TipTop (54 751,3) и FaceBoom (32 445,6). Расходы на рекламу по остальным каналам значительно ниже: в начале анализируемого периода третье место делили WahooNetBanner с (5 151) и AdNonSense, но в июле последний выбыл из соревнования. Последние места делят между собой каналы YRabbit (944,22) и MediaTornado (954,48). Для канала TipTop с мая по сентябрь мы можем наблюдать только рост расходов (что изображено наглядно на графиках и понятно из сводных таблиц). Траты на FaceBoom росли более органично. Расходы на органических клиентов при рассчетах не учитывались, тк не имеют место быть.

Привлечение одного пользователя из канал TipTop обошлось компании в среднем в 2,76, из FaceBoom - 1,11, из AdNonSense - 1,01. Если по большинству каналов расходы на протяжении всего времени были примерно равномерными, то TipTop на их фоне значительно выделяется, т.к. расходы на одного пользователя росли почти каждый месяц. Средний CAC по всему проекту составил около 0,69.

### 3.3  Оцените окупаемость рекламы

Используя графики LTV, ROI и CAC, проанализируйте окупаемость рекламы. Считайте, что на календаре 1 ноября 2019 года, а в бизнес-плане заложено, что пользователи должны окупаться не позднее чем через две недели после привлечения. Необходимость включения в анализ органических пользователей определите самостоятельно.

* Проанализируйте окупаемость рекламы c помощью графиков LTV и ROI, а также графики динамики LTV, CAC и ROI.
* Проверьте конверсию пользователей и динамику её изменения. То же самое сделайте с удержанием пользователей. Постройте и изучите графики конверсии и удержания.
* Проанализируйте окупаемость рекламы с разбивкой по устройствам. Постройте графики LTV и ROI, а также графики динамики LTV, CAC и ROI.
* Проанализируйте окупаемость рекламы с разбивкой по странам. Постройте графики LTV и ROI, а также графики динамики LTV, CAC и ROI.
* Проанализируйте окупаемость рекламы с разбивкой по рекламным каналам. Постройте графики LTV и ROI, а также графики динамики LTV, CAC и ROI.
* Ответьте на такие вопросы:
  + Окупается ли реклама, направленная на привлечение пользователей в целом?
  + Какие устройства, страны и рекламные каналы могут оказывать негативное влияние на окупаемость рекламы?
  + Чем могут быть вызваны проблемы окупаемости?

Напишите вывод, опишите возможные причины обнаруженных проблем и промежуточные рекомендации для рекламного отдела.

Ввод [38]:

*# исключение органических пользователей из таблицы*

profiles **=** profiles.query('channel != "organic"')

Ввод [39]:

*# функция для сглаживания фрейма*

**def** filter\_data(df, window):

*# для каждого столбца применяем скользящее среднее*

**for** column **in** df.columns.values:

df[column] **=** df[column].rolling(window).mean()

**return** df

​

Ввод [40]:

*# функция для визуализации LTV и ROI*

**def** plot\_ltv\_roi(ltv, ltv\_history, roi, roi\_history, horizon, window**=**7):

​

*# задаём сетку отрисовки графиков*

plt.figure(figsize**=**(30, 15))

​

*# из таблицы ltv исключаем размеры когорт*

ltv **=** ltv.drop(columns**=**['cohort\_size'])

*# в таблице динамики ltv оставляем только нужный лайфтайм*

ltv\_history **=** ltv\_history.drop(columns**=**['cohort\_size'])[[horizon **-** 1]]

​

*# стоимость привлечения запишем в отдельный фрейм*

cac\_history **=** roi\_history[['cac']]

​

*# из таблицы roi исключаем размеры когорт и cac*

roi **=** roi.drop(columns**=**['cohort\_size', 'cac'])

*# в таблице динамики roi оставляем только нужный лайфтайм*

roi\_history **=** roi\_history.drop(columns**=**['cohort\_size', 'cac'])[

[horizon **-** 1]

]

*# первый график — кривые ltv*

ax1 **=** plt.subplot(2, 3, 1)

ltv.T.plot(grid**=True**, ax**=**ax1)

plt.legend()

plt.xlabel('Лайфтайм')

plt.title('LTV', fontsize**=**25)

​

*# второй график — динамика ltv*

ax2 **=** plt.subplot(2, 3, 2, sharey**=**ax1)

*# столбцами сводной таблицы станут все столбцы индекса, кроме даты*

columns **=** [name **for** name **in** ltv\_history.index.names **if** name **not** **in** ['dt']]

filtered\_data **=** ltv\_history.pivot\_table(

index**=**'dt', columns**=**columns, values**=**horizon **-** 1, aggfunc**=**'mean'

)

filter\_data(filtered\_data, window).plot(grid**=True**, ax**=**ax2)

plt.xlabel('Дата привлечения')

plt.title('Динамика LTV пользователей на {}-й день'.format(horizon), fontsize**=**25)

​

*# третий график — динамика cac*

ax3 **=** plt.subplot(2, 3, 3, sharey**=**ax1)

*# столбцами сводной таблицы станут все столбцы индекса, кроме даты*

columns **=** [name **for** name **in** cac\_history.index.names **if** name **not** **in** ['dt']]

filtered\_data **=** cac\_history.pivot\_table(

index**=**'dt', columns**=**columns, values**=**'cac', aggfunc**=**'mean'

)

filter\_data(filtered\_data, window).plot(grid**=True**, ax**=**ax3)

plt.xlabel('Дата привлечения')

plt.title('Динамика стоимости привлечения пользователей', fontsize**=**25)

*# четвёртый график — кривые roi*

ax4 **=** plt.subplot(2, 3, 4)

roi.T.plot(grid**=True**, ax**=**ax4)

plt.axhline(y**=**1, color**=**'red', linestyle**=**'--', label**=**'Уровень окупаемости')

plt.legend()

plt.xlabel('Лайфтайм')

plt.title('ROI', fontsize**=**25)

​

*# пятый график — динамика roi*

ax5 **=** plt.subplot(2, 3, 5, sharey**=**ax4)

*# столбцами сводной таблицы станут все столбцы индекса, кроме даты*

columns **=** [name **for** name **in** roi\_history.index.names **if** name **not** **in** ['dt']]

filtered\_data **=** roi\_history.pivot\_table(

index**=**'dt', columns**=**columns, values**=**horizon **-** 1, aggfunc**=**'mean'

)

filter\_data(filtered\_data, window).plot(grid**=True**, ax**=**ax5)

plt.axhline(y**=**1, color**=**'red', linestyle**=**'--', label**=**'Уровень окупаемости')

plt.xlabel('Дата привлечения')

plt.title('Динамика ROI пользователей на {}-й день'.format(horizon), fontsize**=**25)

plt.tight\_layout()

plt.show()

Ввод [41]:

*# считаем LTV и ROI*

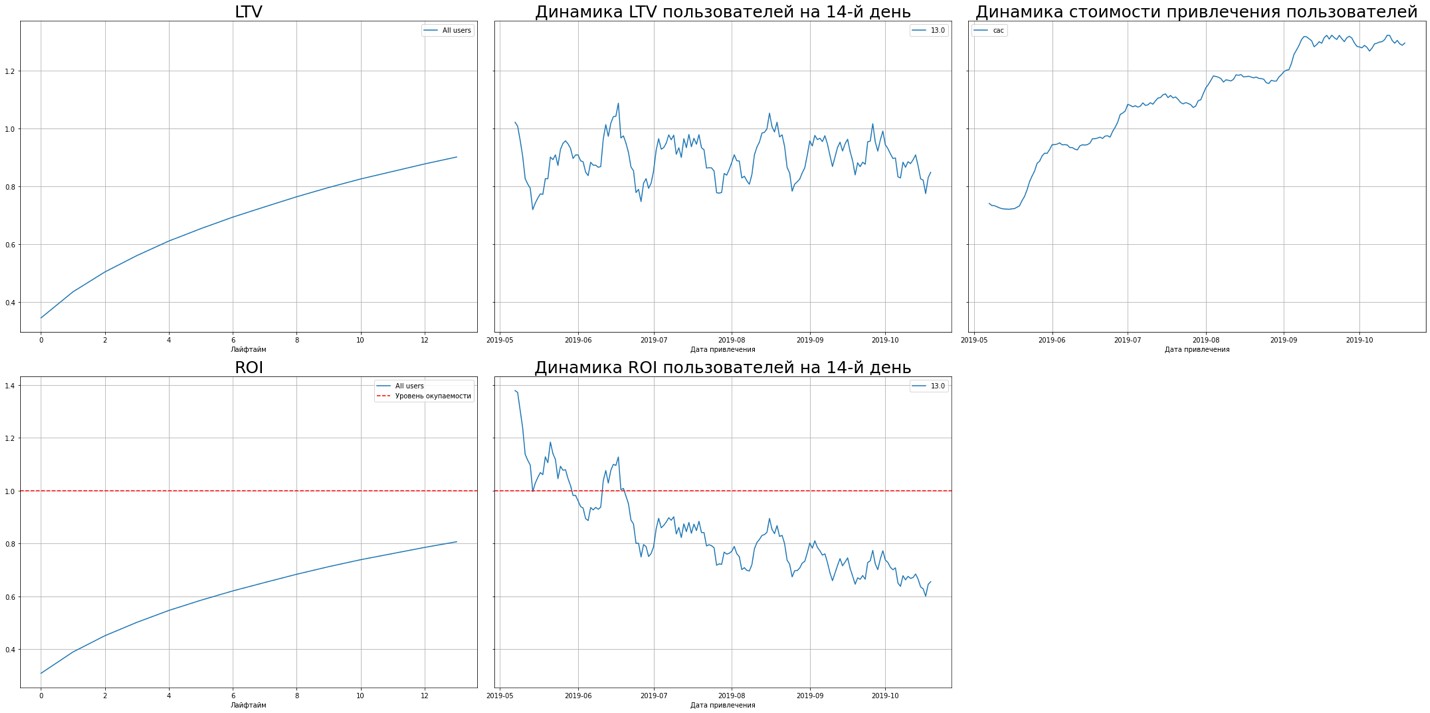
ltv\_raw, ltv\_grouped, ltv\_history, roi\_grouped, roi\_history **=** get\_ltv(

profiles, orders, observation\_date, horizon\_days)

​

*# строим графики*

plot\_ltv\_roi(ltv\_grouped, ltv\_history, roi\_grouped, roi\_history, horizon\_days)



Глядя на графики мы можем видеть, что инвестиции в привлечение покупателей все растут, но не окупаются (ROI планомерно падает в течении всего горизонта событий).

Ввод [42]:

*# считаем LTV и ROI*

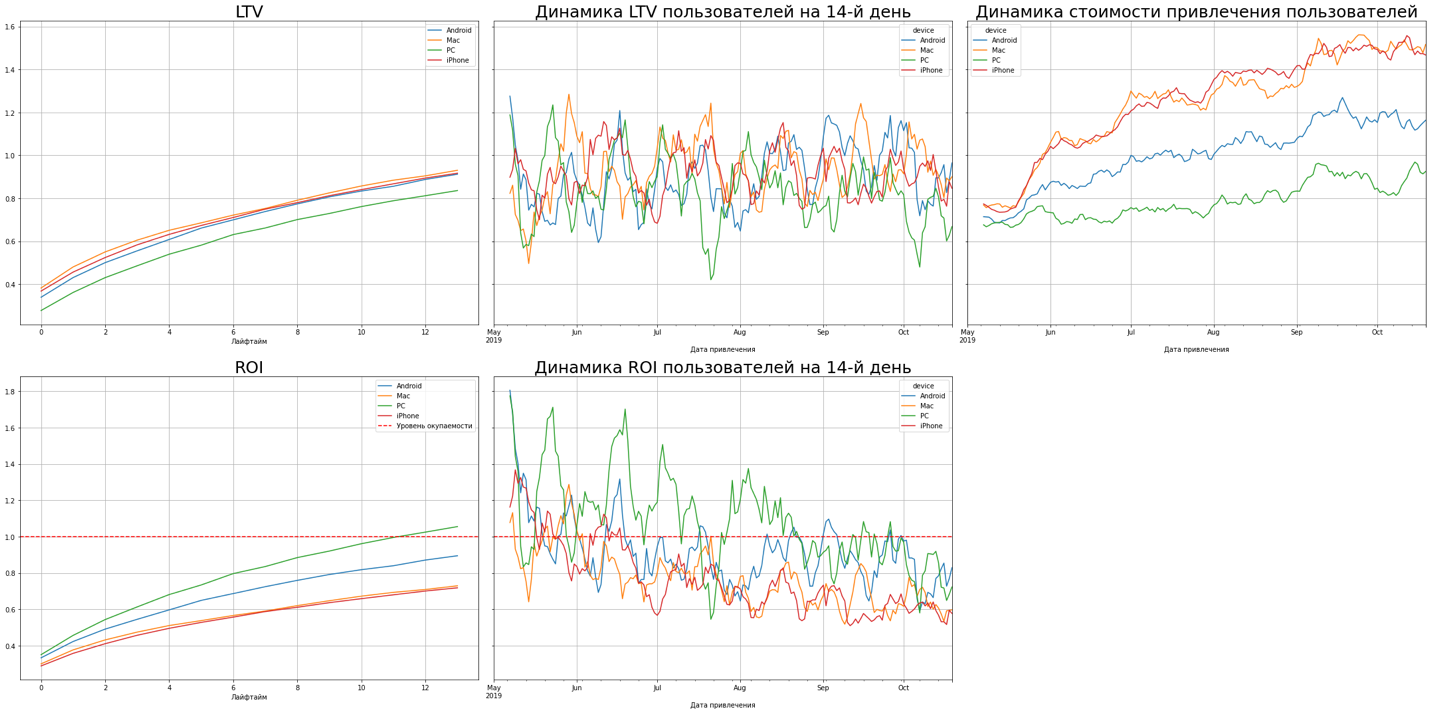
ltv\_raw, ltv\_grouped, ltv\_history, roi\_grouped, roi\_history **=** get\_ltv(

profiles, orders, observation\_date, horizon\_days, dimensions**=**['device'])

​

*# строим графики*

plot\_ltv\_roi(ltv\_grouped, ltv\_history, roi\_grouped, roi\_history, horizon\_days)



По графикам мы видим, что пользователи Mac и iPhone приносят больше всего прибыли, но учитывая размер инвестиций на их привлечение вовосе не окупаются, график ROI показывает, что наиболее привлекательными в плане возврата инвестиций являются как раз пользователи PC и от части пользователи Android. Реклама не окупается. ROI по истечении 2 недель – достигает только 80%. CAC растет, скорее всего, из-за ранее выявленного увеличения расходов TipTop и FaceBoom.

Ввод [43]:

*# считаем LTV и ROI*

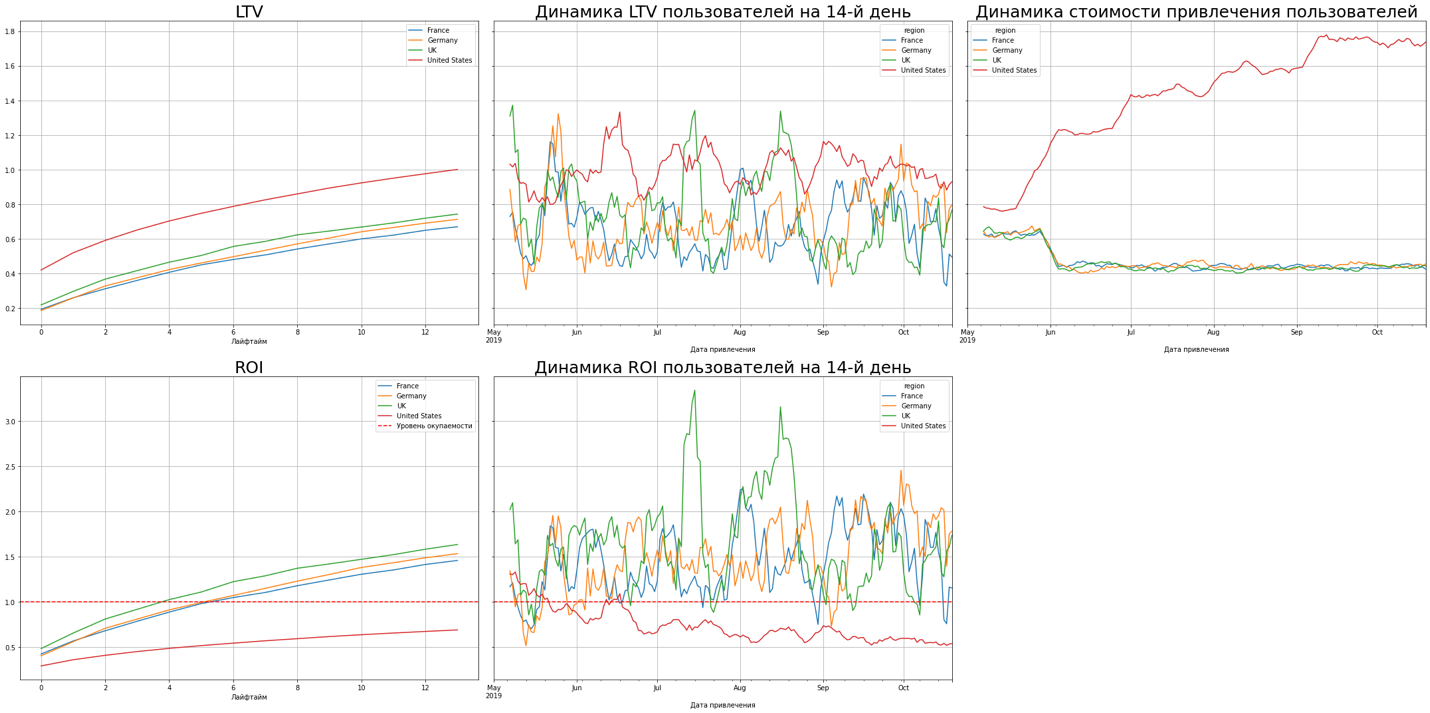
ltv\_raw, ltv\_grouped, ltv\_history, roi\_grouped, roi\_history **=** get\_ltv(

profiles, orders, observation\_date, horizon\_days, dimensions**=**['region'])

​

*# строим графики*

plot\_ltv\_roi(ltv\_grouped, ltv\_history, roi\_grouped, roi\_history, horizon\_days)



CAC значительно вырос среди пользователей из США. Самыми платящими являются пользователи из США (как самая большая группа), но одновременно и стоимость их привлечения выше стоимости привлечения пользователей остальных регионов. Окупается же стоимость привлечения пользователей всех остальных регионов кроме США.

Ввод [44]:

*# считаем LTV и ROI*

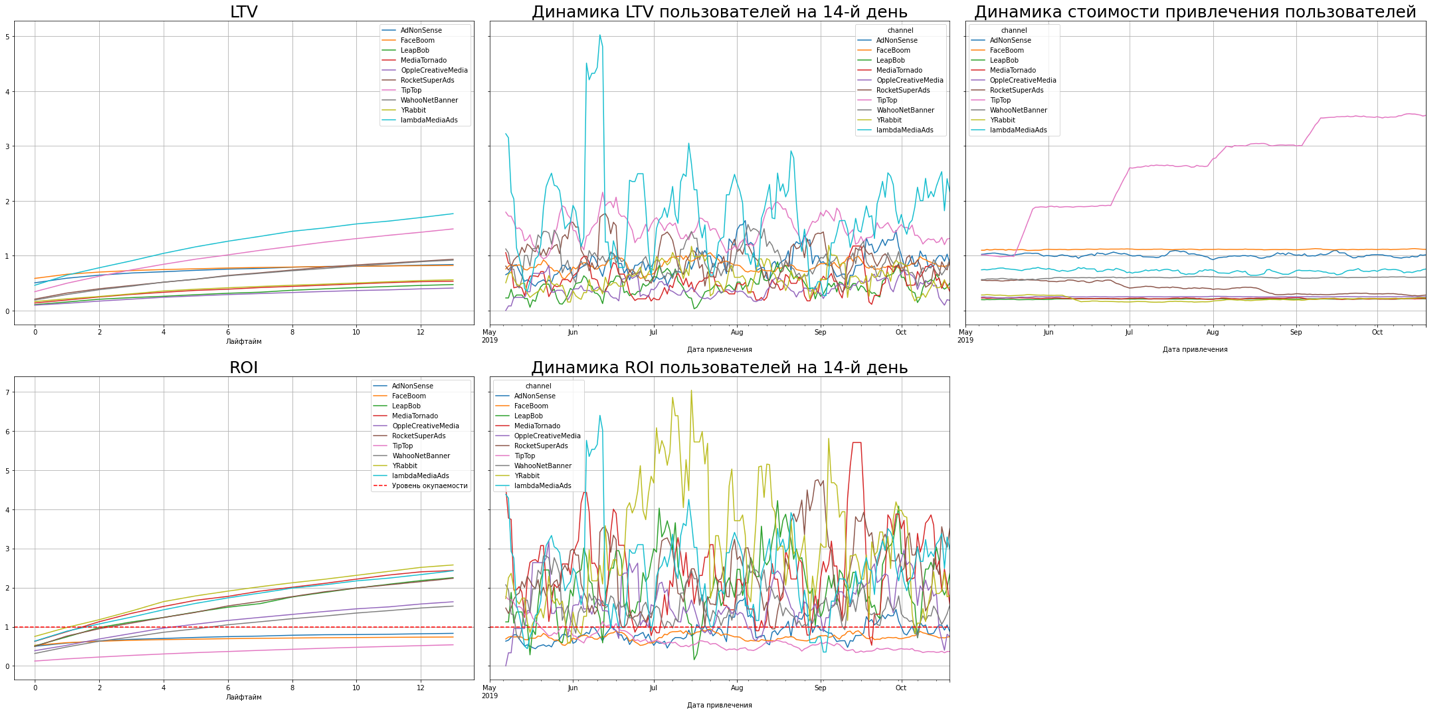
ltv\_raw, ltv\_grouped, ltv\_history, roi\_grouped, roi\_history **=** get\_ltv(

profiles, orders, observation\_date, horizon\_days, dimensions**=**['channel'])

​

*# строим графики*

plot\_ltv\_roi(ltv\_grouped, ltv\_history, roi\_grouped, roi\_history, horizon\_days)



Из 10 каналов только 3 не окупаются, включая TipTop (ROI отрицательный для FaceBoom, AdNonSense и TipTop). Наиболее перспективным и привлекательным с точки зрения показателей выглядит канал lambdaMediaAds: приносит больше всего прибыли при значительно низких расходах на рекламу через него, находится среди лидеров по окупаемости, и меет высокую динамику привлечения пользователей с предсказуемыми пиками.

# Изучение конверсии на графиках

## 4  Функция для визуализации конверсии

Ввод [45]:

**def** plot\_conversion(conversion, conversion\_history, horizon, window**=**7):

​

*# задаём размер сетки для графиков*

plt.figure(figsize**=**(30, 10))

​

*# исключаем размеры когорт*

conversion **=** conversion.drop(columns**=**['cohort\_size'])

*# в таблице динамики оставляем только нужный лайфтайм*

conversion\_history **=** conversion\_history.drop(columns**=**['cohort\_size'])[

[horizon **-** 1]

]

​

*# первый график — кривые конверсии*

ax1 **=** plt.subplot(1, 2, 1)

conversion.T.plot(grid**=True**, ax**=**ax1)

plt.legend()*#bbox\_to\_anchor=(1.0, 1.05), loc='upper left')*

plt.xlabel('Лайфтайм')

plt.title('Конверсия пользователей')

​

*# второй график — динамика конверсии*

ax2 **=** plt.subplot(1, 2, 2, sharey**=**ax1)

columns **=** [

*# столбцами сводной таблицы станут все столбцы индекса, кроме даты*

name **for** name **in** conversion\_history.index.names **if** name **not** **in** ['dt']

]

filtered\_data **=** conversion\_history.pivot\_table(

index**=**'dt', columns**=**columns, values**=**horizon **-** 1, aggfunc**=**'mean'

)

filter\_data(filtered\_data, window).plot(grid**=True**, ax**=**ax2)

plt.xlabel('Дата привлечения')

plt.title('Динамика конверсии пользователей на {}-й день'.format(horizon))

​

plt.tight\_layout()

plt.show()

## 5  Функция для визуализации удержания

Ввод [46]:

**def** plot\_retention(retention, retention\_history, horizon, window**=**7):

​

*# задаём размер сетки для графиков*

plt.figure(figsize**=**(30, 20))

​

*# исключаем размеры когорт и удержание первого дня*

retention **=** retention.drop(columns**=**['cohort\_size', 0])

*# в таблице динамики оставляем только нужный лайфтайм*

retention\_history **=** retention\_history.drop(columns**=**['cohort\_size'])[

[horizon **-** 1]

]

​

*# если в индексах таблицы удержания только payer,*

*# добавляем второй признак — cohort*

**if** retention.index.nlevels **==** 1:

retention['cohort'] **=** 'All users'

retention **=** retention.reset\_index().set\_index(['cohort', 'payer'])

​

*# в таблице графиков — два столбца и две строки, четыре ячейки*

*# в первой строим кривые удержания платящих пользователей*

ax1 **=** plt.subplot(2, 2, 1)

retention.query('payer == True').droplevel('payer').T.plot(

grid**=True**, ax**=**ax1

)

plt.legend()

plt.xlabel('Лайфтайм')

plt.title('Удержание платящих пользователей')

*# в третьей ячейке — динамика удержания платящих*

ax2 **=** plt.subplot(2, 2, 2, sharey**=**ax1)

*# получаем названия столбцов для сводной таблицы*

columns **=** [

name

**for** name **in** retention\_history.index.names

**if** name **not** **in** ['dt', 'payer']

]

*# фильтруем данные и строим график*

filtered\_data **=** retention\_history.query('payer == True').pivot\_table(

index**=**'dt', columns**=**columns, values**=**horizon **-** 1, aggfunc**=**'mean'

)

filter\_data(filtered\_data, window).plot(grid**=True**, ax**=**ax2)

plt.xlabel('Дата привлечения')

plt.title(

'Динамика удержания платящих пользователей на {}-й день'.format(

horizon

)

)

plt.tight\_layout()

plt.show()

Ввод [47]:

*# конверсия с разбивкой по устройствам*

dimensions **=** ['device']

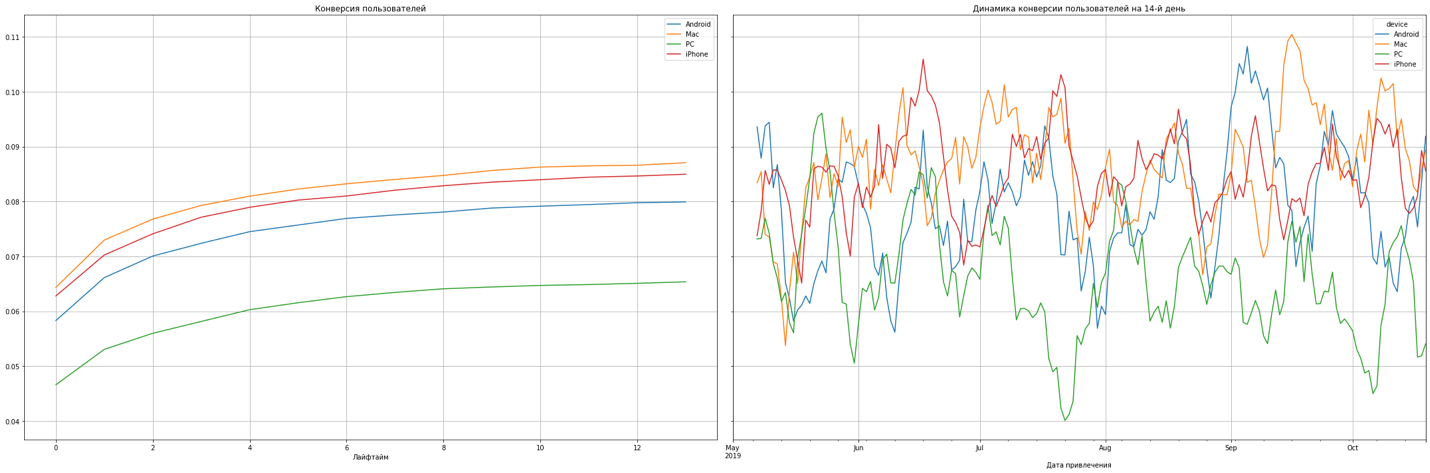
conversion\_raw, conversion\_grouped, conversion\_history **=** get\_conversion(

profiles, orders, observation\_date, horizon\_days, dimensions**=**dimensions

)

​

plot\_conversion(conversion\_grouped, conversion\_history, horizon\_days)



Ввод [48]:

*# удержание с разбивкой по устройствам*

dimensions **=** ['device']

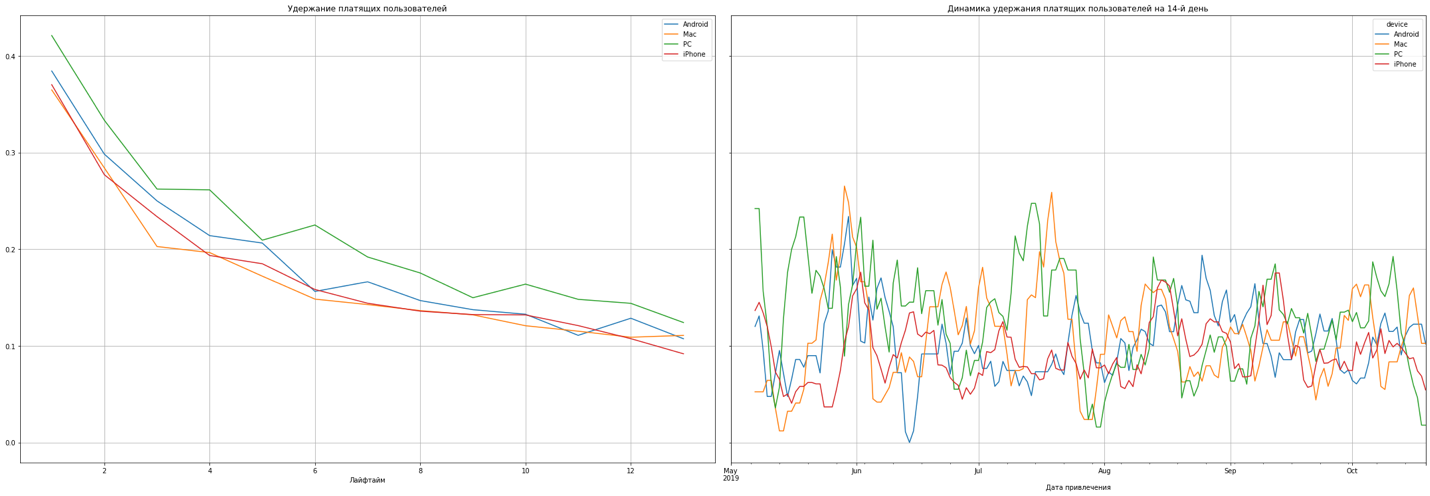
retention\_raw, retention\_grouped, retention\_history **=** get\_retention(

profiles, visits, observation\_date, horizon\_days, dimensions**=**dimensions

)

​

plot\_retention(retention\_grouped, retention\_history, horizon\_days)



Mac и iPhone имеют лучшую конверсию (это гаджеты которыми пользуется большинство клиентов из США - так что по отношению к общей конверсии - неоднозначный результат). Пользователи же РС показывают лучше результат удержания.

Ввод [49]:

*# конверсия с разбивкой по регионам*

dimensions **=** ['region']

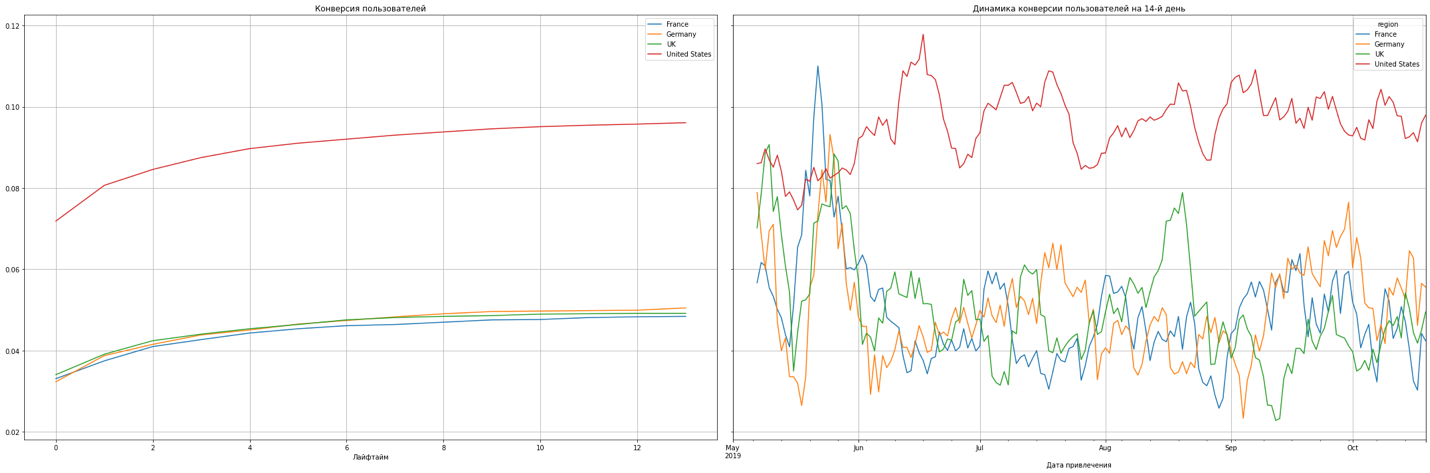
conversion\_raw, conversion\_grouped, conversion\_history **=** get\_conversion(

profiles, orders, observation\_date, horizon\_days, dimensions**=**dimensions

)

​

plot\_conversion(conversion\_grouped, conversion\_history, horizon\_days)



Ввод [50]:

*# удержание с разбивкой по регионам*

dimensions **=** ['region']

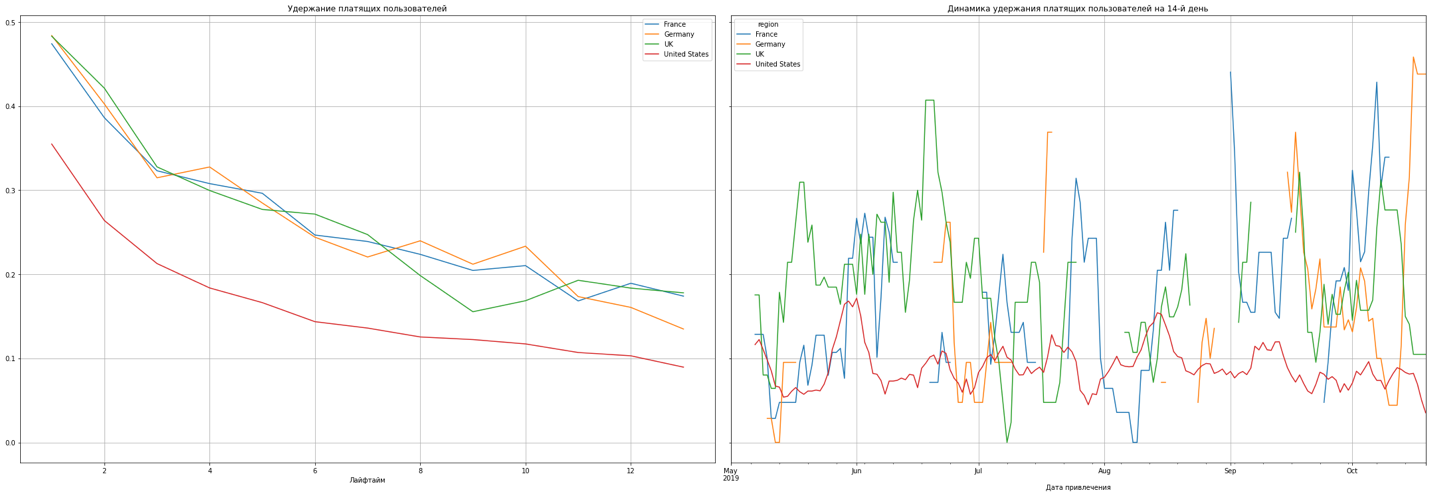
retention\_raw, retention\_grouped, retention\_history **=** get\_retention(

profiles, visits, observation\_date, horizon\_days, dimensions**=**dimensions

)

​

plot\_retention(retention\_grouped, retention\_history, horizon\_days)



Пользователи США почти в 2 раза опережают по конверсии (но мы-то помним их ROI) и уступают по удержанию.

Ввод [51]:

*# конверсия с разбивкой по каналам*

dimensions **=** ['channel']

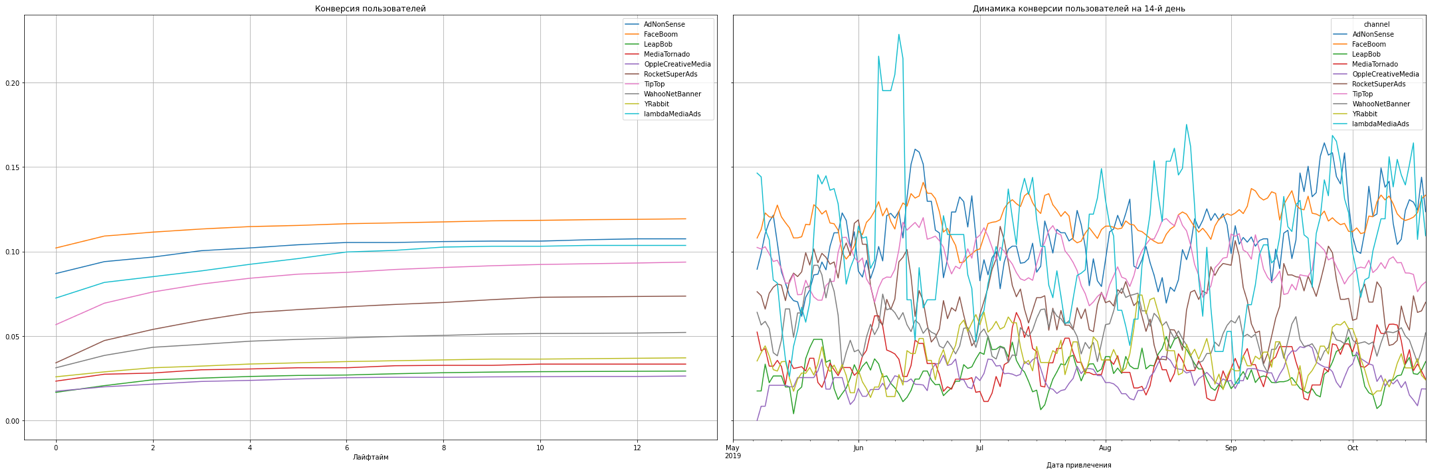
conversion\_raw, conversion\_grouped, conversion\_history **=** get\_conversion(

profiles, orders, observation\_date, horizon\_days, dimensions**=**dimensions

)

​

plot\_conversion(conversion\_grouped, conversion\_history, horizon\_days)



Ввод [52]:

*# удержание с разбивкой по каналам*

dimensions **=** ['channel']

retention\_raw, retention\_grouped, retention\_history **=** get\_retention(

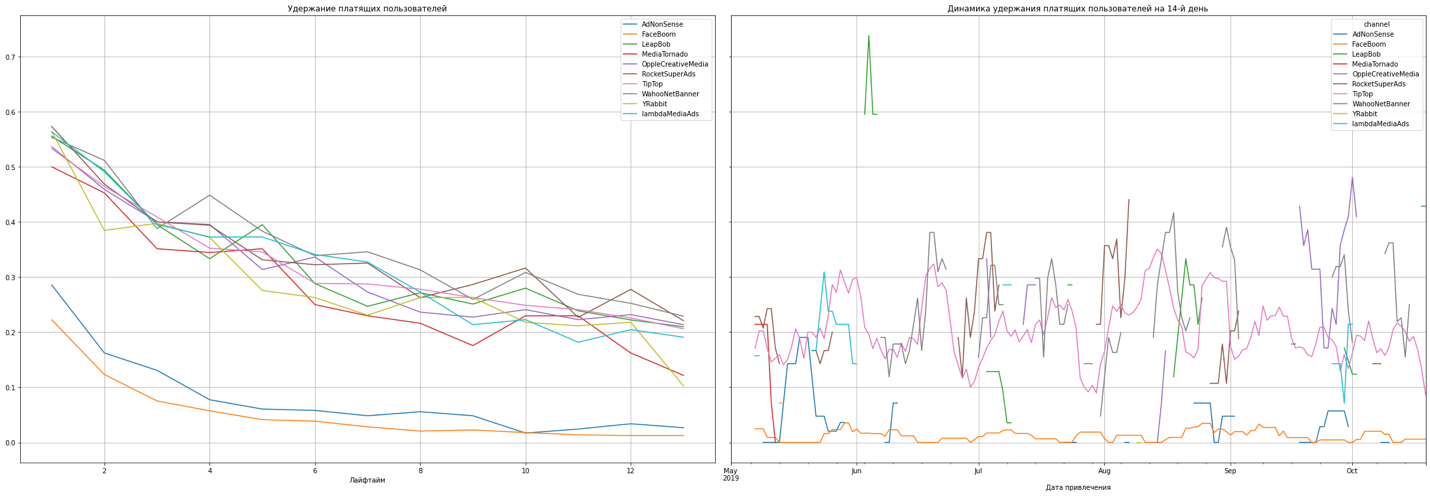
profiles, visits, observation\_date, horizon\_days, dimensions**=**dimensions

)

​

plot\_retention(retention\_grouped, retention\_history, horizon\_days)

​



Удержание большинства равномерно, кроме клиентов от FaceBoom, при высокой конверсии. Также одним из худших по удержанию оказался рекламный источник AdNonSense.

Отвечая на поставленные вопросы хочу заметить, что в целом реклама себя не окупила. Было инвестировано большое количество средств в маркетинг для пользователей США, которые и оказались самыми дорогими и при этом самыми не рентабильными - думаю, что эта большая группазначительно искажает ислледование рентабильности приложения только за счет своего размера. Возможно на стоимость рекламы в США оказывают большое влияние экономическая и политическая обстановки, которые крайне важн анализировать, прежде, чем инвестировать в тот или иной регион. А так как нам удалось убедиться в ходе анализа, что все регионы, кроме США, имеют гораздо больший коэффициент окупаемости инвестиций и их динамика выглядит куда стабильнее - могу предложить уделить больше внимания разработке специальных предложений именно для этих регионов с учетом из политической и экономической обстановки.

### 5.1  Напишите выводы

* Выделите причины неэффективности привлечения пользователей.
* Сформулируйте рекомендации для отдела маркетинга.

# Выводы.

В исследовании проанализированы данные по пользователям приложения Procrastinate Pro+ из США, Великобритании, Франции и Германии. Загружены и получены данные, проведен их исследовательский анализ, загружены функции для анализа ключевых бизнес показателей, проведен анализ маркетинга и визуализированы результаты,оценена окупаемость рекламы. Предлагаю вам ознакомиться с общими выводами по проекту.

США составляет наибольшую группу из анализируемых пользователей, в т.ч. платящих (всего примерно 100 тыс. чел.). iPhonе лидирует среди устройств, как гаджет, которым чаще всего пользуются пользователи самой крупной группы пользователей -жителей США. TipTop и FaceBoom являются каналами, в рекламу по которым было вложено наибольшее количество средств на маркетинг. Например, привлечение одного пользователя из канала TipTop обошлось компании в среднем в 2,76, из FaceBoom - 1,11, при этом на TipTop расходы на одного пользователя росли почти каждый месяц (видимо в соответствии с рекламной стратегией на анализируемый период).

При анализе расходов и окупаемости в разрезе по странам, устройствам и каналам выявлено, что стратегия инвестировать в рекламу для пользователей из США была ошибочной. К причинам неэффективности рекламы отнесем:

* у пользователей приложений больше всего пользователей из США, в то время как рентабельность инвестиций в рекламу самая низкая для пользователей из США, и удержание среди пользователей из США также страдает. За рассматриваемый период CAC пользователей из США удвоился, в основном из-за рекламы в США на TipTop;
* реклама AdNonSense так же не акупается и не эффекитивна;
* низкая конверсия среди пользователей ПК;
* низкое удержание пользователей, которые нажимают на рекламу от FaceBoom и AdNonSense.

Возможно на стоимость рекламы в США оказывают большое влияние экономическая и политическая обстановки, которые крайне важно анализировать, прежде, чем инвестировать в тот или иной регион. А так как нам удалось убедиться в ходе анализа, что все регионы, кроме США, имеют гораздо больший коэффициент окупаемости инвестиций и их динамика выглядит куда стабильнее - могу предложить уделить больше внимания разработке специальных предложений именно для этих регионов с учетом из политической и экономической обстановки - такова одна из моих рекоммендаций для отдела маркетинга. Так же, в разрезе по устройствам расходы окупились по пользователям PC, в разрезе по странам - из Франции, Германии и Великобритании.

Проблемы окупаемости вызваны необдуманной стратегией проведения рекламных кампаний: очевидно, что маркетологи ориентировались на показатели конверсии и исходили именно из них. Именно наиболее конвертируемые пользователи, привлекаемые через AdNonSense,FaceBoom и TipTop, пробили существенную дыру в бюджете компании. Они не во всем были неправы: входящая в тройку по уровню конверсии lambdaMediaAds оказалась вполне удачным каналом рекламы. Однако, если им и удалось таким образом поднять общий уровень LTV, то этого оказалось явно недостаточно для того чтобы отбить высокие и постоянно растущие расходы на рекламу. TipTop оказась просто несоразмерно дорогой по привлечению, FaceBoom оказалась дорогой и с низким удержанием, AdNonSense с низким LTV и низким удержанием.

Очевидными причинами неэффективности рекламы можно считать необоснованные вложения рекламного бюджета в источники более качественных новых пользователей при том, что дельта в качестве не способна была перекрыть избыточные затраты на привлечение. Это, по-видимому, произошло из-за ориентации маркетологов исключительно на показатель конверсии. Отдельного упоминания заслуживает тот факт, что этот кошмар длился полгода, хотя никто не мешал обнаружить проблему по прошествии 2-3 месяцев. Иногда стоит следить не только за доходами, но еще и за тратами.

# Рекомендации для отдела маркетинга:

* уменьшить горизонт планирования бюджетов и, соответственно, обеспечить более регулярный анализ бизнес метрик.
* разорвать контракты с AdNonSense,FaceBoom и TipTop. При желании можно предложить этим платформам иные расценки с учетом LTV привлекаемых пользователей и прочих метрик. Кроме того, важно понимать насколько маркетологи оплачивали таргетацию в каналах. Если таргетация оплачивалась и по ip страны привлечения, и по устройству, мы бы рекомендовали снизить затраты, отказавшись от таргетации. При более детальном изучении проблем с неокупаемостью рекламы обнаружилось, что каналы привлечения TipTop и FaceBoom в США и канал AdNonSense в Европе являются корнем зла: денег на их использование уходит немеряно, а пользы от них на порядок меньше. Отказавшись от вложений в рекламу на этих каналах компания сможет выходить в окупаемость к 3-ему лайфтайму.

# Альтернативные рекомендации для отдела маркетинга:

* для США можжно обратить внимание на каналы RocketSuperAds и YRabbit. У RocketSuperAds хорошо растет LTV и падает CAC. YRabbit имеет самый высокий ROI (но канал имеет высокую амплитуду колебаний и необходимо выяснить в связи с чем);
* по странам Европы у lambdaMediaAds лучшие показатели LTV и ROI (это один из наиболее успешных каналов рекламы и в моем понимании маркетологам стоило бы выяснить как выправить подобную амплитуду и добиться более ровного качества привлекаемых пользователей, желательно по верхней планке, естественно). В остальном, качество на отрезке 6 мес в среднем выглядит более или менее стабильно. Канал имеет высокия показания конверсии.