Анализ бизнес-показателей приложения ProcrastinatePRO+

Заказчик: отдел маркетинга развлекательного приложения Procrastinate Pro+

Описание задачи проекта: несмотря на огромные вложения в рекламу, последние несколько месяцев компания терпит убытки. Задача — выявить причины неэффективности привлечения пользователей и сформировать рекомендации для отдела маркетинга для повышения эффективности рекламы.

Входные данные:

- visits_info_short - данные о визитах пользователей
- orders_info_short - данные о заказах
- visits_info_short - данные о расходах на рекламу

Навыки и инструменты, применённые в работе:

- **Предобработка данных:** переименование столбцов, замена типов данных, проверка на дубликаты, проверка на пропуски, проверка данных на соответствие периоду, заявленному в ТЗ.
- **Расчёты и исследование данных:**

- функция для создания пользовательских профилей
- функция для расчёта удержания
- функция для расчёта конверсии
- функция для расчёта LTV и ROI
- функция для сглаживания фрейма (применение для каждого столбца скользящего среднего)
- функции для визуализации удержания, конверсии,LTV и ROI

- **Графики:**

- Динамика распределения суммарных расходов на рекламу по каналам привлечения,
- Количество затрат по месяцам по каждому каналу
- Рассчитаем средний CAC на одного пользователя, для каждого источника трафика
- Динамика CAC за весь период, по каналам привлечения и т.д.

Ход исследования:

Шаг1. Загружаем и подготовливаем полученные данные к анализу.

Шаг2. Задаём функции для расчета и анализа LTV, ROI, удержания и конверсии.

Шаг3. Проведём исследовательский анализ данных:

- Создаём пользовательские профили.
- Выясняем из каких стран пользователи приходят в приложение, какими устройствами пользуются клиенты и какие устройства предпочитают платящие пользователи, рекламные источники привлечения и определите каналы, из которых пришло больше всего платящих пользователей.

Шаг4. Считаем общую сумму расходов на маркетинг. Выясныем, как траты распределены по источникам. Узнаем, сколько в среднем стоило привлечение одного пользователя из каждого источника. Считаем средний CAC на одного пользователя для всего проекта и для каждого источника трафика.

Шаг 5. Оцениваем окупаемость рекламы.

Используя графики LTV, ROI и CAC, проанализируем окупаемость рекламы исходя из горизонта 14 дней. Органических пользователей не включаем в рассчёты, т.к. на их привлечение прямых расходов не было.

Описание данных

Таблица visits_log_short (лог сервера с информацией о посещениях сайта):

```
User Id – уникальный идентификатор пользователя
Device – категория устройства пользователя
Session start – дата и время начала сессии
Session End – дата и время окончания сессии
Channel – идентификатор рекламного источника, из которого пришел пользователь
Region - страна пользователя
```

Таблица orders_log_short (информация о заказах):

```
User Id — уникальный id пользователя, который сделал заказ
Event Dt — дата и время покупки
Revenue — выручка
```

Таблица costs_short (информация о затратах на маркетинг):

```
Channel — идентификатор рекламного источника
Dt — дата
Costs — затраты на этот рекламный источник в этот день
```

Содержание

Шаг 1. Загрузка данных и подготовка их к анализу

Импортируем нужные библиотеки, загрузим данные(журнала посещений,журнала покупок,журнала расходов на рекламу) и преобразуем значения в столбцах.

In [19]:
```
import pandas as pd
from datetime import datetime, timedelta
```

```
import seaborn as sns
from matplotlib import pyplot as plt
import numpy as np
```

In [20]:
```
visits = pd.read_csv('visits_info_short.csv')
orders = pd.read_csv('orders_info_short.csv')
costs =  pd.read_csv('costs_info_short.csv')
```

In [21]:
```
visits.head(3)
```

Out[21]:

	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

```
<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
```

Out[23]:

	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

In [24]:
```
# доработаем название столбцов:
visits.rename(columns = {'User Id':'user_id','Region':'region', 'Device':'device', 'Channel':'channel','Session S
```

In [25]:
```
# преобразуем формат данных в колонках Session Start и Session End с типа object на datetime для дальнейшей работ
visits['session_start'] = pd.to_datetime(visits['session_start'])
visits['session_end'] = pd.to_datetime(visits['session_end'])
```

In [26]:
```
# проверим наличие дубликатов с учётом регистра
visits['region'] = visits['region'].str.lower()
visits['device'] = visits['device'].str.lower()
visits['channel'] = visits['channel'].str.lower()
visits.duplicated().sum()
```

Out[26]:
```
0
```

```
<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
```

Out[29]:

	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

In [30]:
```
orders.rename(columns = {'User Id':'user_id', 'Event Dt':'event_dt', 'Revenue':'revenue'}, inplace = True)
```

In [31]:
```
# преобразуем формат данных в колонке Event Dt с типа object на datetime для дальнейшей работы с ним:
orders['event_dt']=pd.to_datetime(orders['event_dt'])
```

In [32]:
```
# проверим наличие дубликатов
orders.duplicated().sum()
```

Out[32]:
```
0
```

In [33]:
```
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  datetime64[ns]
 2   revenue   40212 non-null  float64       
dtypes: datetime64[ns](1), float64(1), int64(1)
memory usage: 942.6 KB
```

In [34]:
```
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
```

In [35]:
```
costs.head()
```

Out[35]:

	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

In [36]:
```
costs.rename(columns = {'Channel' : 'channel'}, inplace = True)
```

In [37]:
```
# преобразуем формат данных в колонке dt с типа object на datetime для дальнейшей работы с ним:
costs['dt'] = pd.to_datetime(costs['dt']).dt.date
```

In [38]:
```
# проверим наличие дубликатов с учётом регистра
costs['channel'] = costs['channel'].str.lower()
costs.duplicated().sum()
```

```
0
```

Out[38]: 0

In [39]:
```
#costs.info()
```

Выводы по качеству данных: по результатам проверки загруженных данных во всех трёх таблицах были скорректированы названия столбцов: приведены к нижнему регистру и "змеиному стилю", столбцы, содержащие информацию о дате и времени были приведены к типу datetime, а также во всех таблицах сделана проверка на дубликаты и пропуски - их не обнаружено. По итогом предобработки данных, можно сказать, что данные достаточно хорошего качества.

Добавим по каждой таблице проверки по тому, насколько данные соответсвуют исследуемому периоду:

```
Количество строк до 5 мая 2019г. в таблице visits: 0
Количество строк после 27 октября 2019г. в таблице visits: 5972
Максимальная дата записи в таблице visits: 2019-11-01 01:38:46

Количество строк до 5 мая 2019г. в таблице orders: 0
Количество строк после 27 октября 2019г. в таблице orders: 1427
Количество строк после 27 октября 2019г. в таблице orders: 2019-10-31 23:56:56

Минимальная дата записи в таблице costs: 2019-05-01
Максимальная дата в таблице costs: 2019-10-27
```

Далее добавлен столбец длительности сессий и найдена минимальная длина сессии, таких нулевых сессий 163 - это незначительное количество относительно всех данных (309901).

In [41]:
```
visits['duration'] = visits['session_end'] - visits['session_start']
visits['duration'].min()
```

Out[41]: Timedelta('0 days 00:00:00')

In [42]:
```
visits[visits['duration']=='0 days 00:00:00'].count()
```

Out[42]:
```
user_id          163
region           163
device           163
channel          163
session_start    163
session_end      163
duration         163
dtype: int64
```

Проверим таблицы на пропуски в столбцах:

```
Количество пропусков в столбцах таблицы visits:

user_id 0
region 0
device 0
channel 0
session_start 0
session_end 0
duration 0
```

```
Количество пропусков в столбцах таблицы orders:

user_id 0
event_dt 0
revenue 0
```

```
Количество пропусков в столбцах таблицы costs:

dt 0
channel 0
costs 0
```

Шаг 2. Функции для расчета и анализа LTV, ROI, удержания и конверсии

Функция для создания пользовательских профилей get_profiles()

In [46]:

```
def get_profiles(sessions, orders, ad_costs):
    # Шаг 1. Передадим в функцию расчета профиля данные о рекламных затратах (фрейм ad_costs)

    # сортируем сессии по id пользователя и дате для того,чтобы работал first
    # находим первые значения для параметров пользователя - будем считать их основными
    profiles = (sessions.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()  # вернем все данные из индекса в колонки
                )
    # определим дату первого посещеня
    # и начало месяца первого посещения - они понадобятся нам при когортном анализе
    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())

    # Шаг 2. К данным о рекламных затратах добавим количества привлеченных пользователей
    new_users = profiles.groupby(['dt', 'channel']).agg({'user_id': 'nunique'}).rename(columns = {'user_id': 'uni
    ad_costs = ad_costs.merge(new_users, on = ['dt', 'channel'], how = 'left')

    # Шаг 3. Найдем среднюю стоимость привлечения пользователя
    ad_costs['acquisition_cost'] = ad_costs['costs'] / ad_costs['unique_users']

    # Шаг 4. Присоединим данные к профилям пользователей информацию о средней стоимости привлечения в день привле
    profiles = profiles.merge(ad_costs[['dt', 'channel', 'acquisition_cost']], on = ['dt', 'channel'], how = 'lef
    profiles['acquisition_cost'] = profiles['acquisition_cost'].fillna(0) # органические пользователи будут стои

    return profiles
```

Функция для расчёта удержания get_retention()

In [47]:

```
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
```

Функция для расчёта конверсии get_conversion()

In [48]:

```
# функция для расчёта конверсии

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
```

Функция для расчёта LTV и ROI get_ltv()

In [49]:

```
# функция для расчёта LTV и ROI

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
    )
```

Функция для сглаживания фрейма

In [50]:

```
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
```

Функция для визуализации удержания plot_retention()

In [51]:

```
def plot_retention(retention, retention_history, horizon, window=7):

    # задаём размер сетки для графиков
    plt.figure(figsize=(15, 10))

    # 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')

    # во второй ячейке строим кривые удержания неплатящих
    # вертикальная ось — от графика из первой ячейки
    ax2 = plt.subplot(2, 2, 2, sharey=ax1)
    retention.query('payer == False').droplevel('payer').T.plot(
        grid=True, ax=ax2
    )
    plt.legend()
    plt.xlabel('Лайфтайм')
    plt.title('Удержание неплатящих пользователей')

    # в третьей ячейке — динамика удержания платящих
    ax3 = plt.subplot(2, 2, 3)
    # 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=ax3)
    plt.xlabel('Дата привлечения')
    plt.title(
        'Динамика удержания платящих пользователей на {}-й день'.format(
            horizon
        )
    )

    # в чётвертой ячейке — динамика удержания неплатящих
    ax4 = plt.subplot(2, 2, 4, sharey=ax3)
    # фильтруем данные и строим график
    filtered_data = retention_history.query('payer == False').pivot_table(
        index='dt', columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax4)
    plt.xlabel('Дата привлечения')
    plt.title(
        'Динамика удержания неплатящих пользователей на {}-й день'.format(
            horizon
        )
    )

    plt.tight_layout()
    plt.show()
```

Функция для визуализации конверсии plot_conversion()

In [52]:

```
def plot_conversion(conversion, conversion_history, horizon, window=7):

    # задаём размер сетки для графиков
    plt.figure(figsize=(15, 5))

    # 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()
    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()
```

Функция для визуализации LTV и ROI plot_ltv_roi()

In [53]:

```
def plot_ltv_roi(ltv, ltv_history, roi, roi_history, horizon, window=7):

    # задаём сетку отрисовки графиков
    plt.figure(figsize=(20, 10))

    # 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')

    # 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))

    # 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')

    # 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')

    # 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))

    plt.tight_layout()
    plt.show()
```

Шаг 3. Исследовательский анализ данных

Профили пользователей

Построим профили пользователей:

Out[54]:

	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

Определим минимальную и максимальную дату привлечения пользователей:

In [55]:
```
min_analysis_date = profiles['first_ts'].min()
max_analysis_date = profiles['first_ts'].max()

print(min_analysis_date)

print(max_analysis_date)
```

```
2019-05-01 00:00:41
2019-10-27 23:59:04
```

Минимальная и максимальная дата в таблице с профилями привлеченных клиентов соответсвует временным рамкам в тех.задании

Страны

Выясним, из каких стран приходят посетители и какие страны дают больше всего платящих пользователей:

Out[56]:

	region	user_id
0	united states	100002
1	uk	17575
2	france	17450
3	germany	14981

Найдем по каждой стране % платящих пользователей от их общего числа:

Out[57]:

	region	user_id	payer
0	united states	100002	6.90
1	germany	14981	4.11
2	uk	17575	3.98
3	france	17450	3.80

Устройства

Выясним,какими устройствами пользуются посетители и с каких устройств чаще всего заходят платящие пользователи:

Out[58]:

	device	user_id
0	iphone	54479
1	android	35032
2	pc	30455
3	mac	30042

Найдем по каждому устройству % платящих пользователей от их общего числа:

Out[59]:

	device	user_id	payer
0	mac	30042	6.36
1	iphone	54479	6.21
2	android	35032	5.85
3	pc	30455	5.05

Каналы привлечения

Выясним, по каким рекламным каналам шло привлечение пользователей и какие каналы приносят больше всего платящих пользователей:

Out[60]:

	channel	user_id
0	organic	56439
1	faceboom	29144
2	tiptop	19561
3	opplecreativemedia	8605
4	leapbob	8553
5	wahoonetbanner	8553
6	rocketsuperads	4448
7	mediatornado	4364
8	yrabbit	4312
9	adnonsense	3880
10	lambdamediaads	2149

Найдем по каждому каналу % платящих пользователей от их общего числа:

Out[61]:

	channel	user_id	payer
0	faceboom	29144	12.20
1	adnonsense	3880	11.34
2	lambdamediaads	2149	10.47
3	tiptop	19561	9.60
4	rocketsuperads	4448	7.91
5	wahoonetbanner	8553	5.30
6	yrabbit	4312	3.83
7	mediatornado	4364	3.57
8	leapbob	8553	3.06
9	opplecreativemedia	8605	2.71
10	organic	56439	2.06

Промежуточные выводы

Страны

По количеству привлечённых пользователей лидирует США, при этом 69% из них платящие. На втором месте находится Великобритания, но по доле платящих пользователей (почти 40%) - находится на 3-м месте. На третьем месте - Франция, но при этом по доле платящих (38%) она на 4-м месте. На четвертом месте - Германия, но после США у неё самое большая доля платящих пользователей (41,1%)

Устройства

Чаще всего пользователи заходят с iphone, при этом с доля платящих пользователей также очень высока (62,1%) - это чуть ниже, чем у mac (63,6% доля платящих пользователей), которые при этом находятся последнем, четвертом месте по количесту заходов с них пользователей.

На втором месте по кол-ву пользователей находятся устройства android, но по доле платящих пользователей (58,5%) - они на 3-м месте.

На третьем месте по кол-ву пользователей находятся десктопные устройства, по доле платящих пользователей (50,5%) - они на последнем, 4-м месте.

Каналы привлечения

Пользователи приходят из 10 каналов, в том числе органический. В пятёрку лидеров по кол-ву привлеченных пользователей входят: organic(56439), faceboom(29144), tiptop(19561), opplecreativemedia (8605), leapbob(8553) и wahoonetbanner(8553) (одинаковое кол-во пользователей).

По доле платящих пользователей в пятерке лидеров есть отличия с пятёркой по количеству привлечённых пользователей: faceboom (12,2%), adnonsense (11,3%), lambdamediaads (10,4 %), tiptop (9,6%), rocketsuperads (7,9%)

В данном разделе мы посмотрели на базовые значения конверсии, доли органики и покупатели по регионам

В в данном разделе мы посмотрели на базовые значения конверсии пользователеи в покупатели по регионам, устройствам и каналу привлечения, определили основной рынок. Получается, что больше всего приходит пользователей из США и они лучше других конвертируется. При этом большая часть пользователей заходит с мобильных устройств. Также, можно сказать, что наибольшую конверсию имеют пользователи, которые пользуются Mac, а затем следуют пользователи iPhone, т.е.в целом пользователи Apple имеют лучшую конверсию в покупателей. Возможно, тут есть плюсы ApplePay.

Шаг 4. Маркетинг

Выясним, сколько всего потратили денег на рекламу:

In [62]:
```
sum_costs = costs['costs'].sum()
sum_costs.round()
```

Out[62]: 105497.0

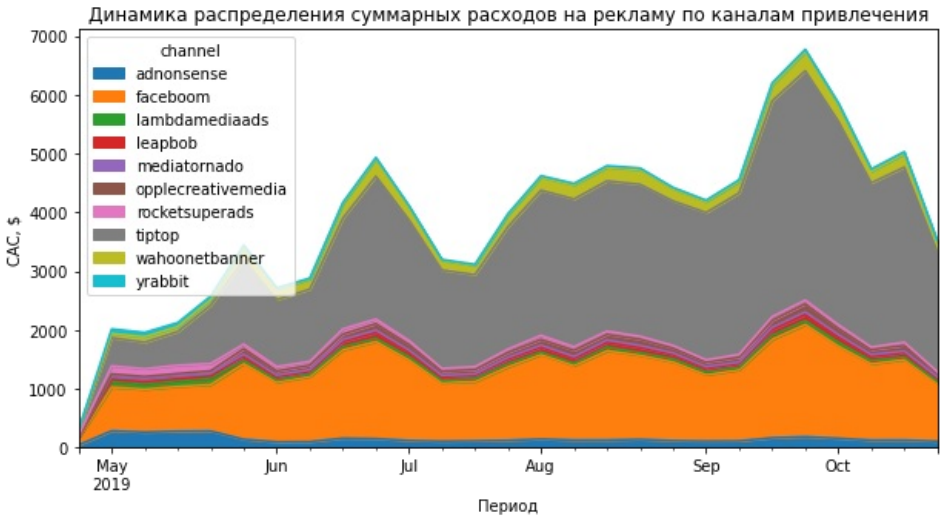
Посчитаем, сколько потратили на каждый из источников:

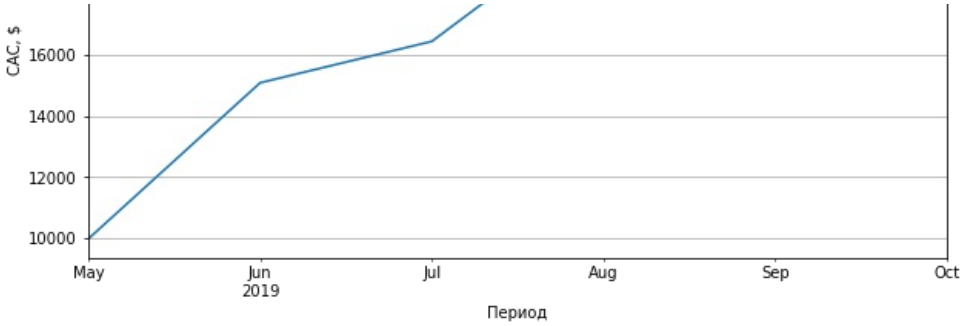
Out[63]:

	channel	acquisition_cost
0	tiptop	54751.30
1	faceboom	32445.60
2	wahoonetbanner	5151.00
3	adnonsense	3911.25
4	opplecreativemedia	2151.25
5	rocketsuperads	1833.00
6	leapbob	1797.60
7	lambdamediaads	1557.60
8	mediatornado	954.48
9	yrabbit	944.22
10	organic	0.00

Отобразим распределение расходов на рекламу по каждому источнику по датам:

<Figure size 1800x1080 with 0 Axes>





По графику видим, что расходы на рекламу с мая по август включительно выросли примерно в 2,2 раза, и только в сентябре стали снижаться.

```
<Figure size 2520x1080 with 0 Axes>
```

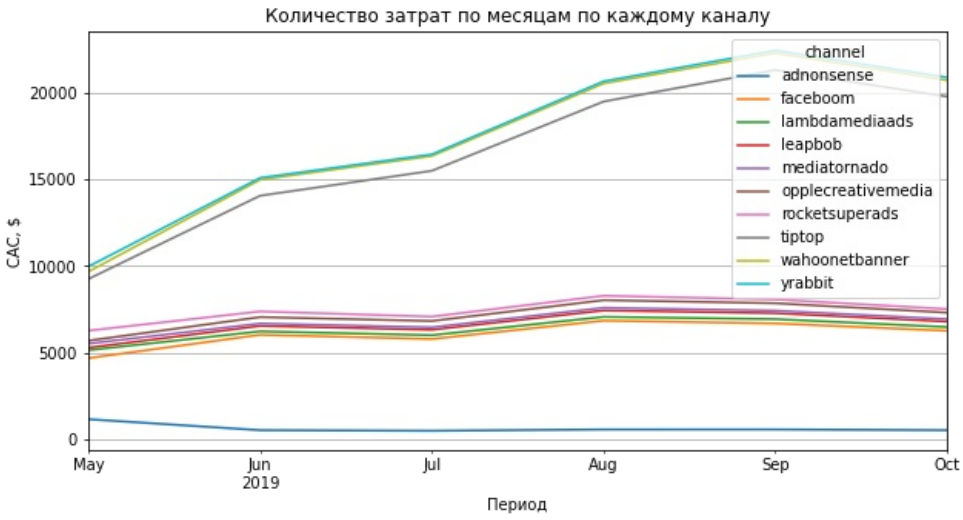


График "Количество затрат по месяцам по каждому каналу" объясняет с чем связан рост расходов на рекламу: по трём рекламным yrabbit, wahoonetbanner, tiptop - аналогично шло увеличение расходов.

Рассчитаем средний CAC на одного пользователя для всего проекта:

In [68]:
```
#отфильтруем профили по признаку "канал привлечения"
#- удалим клиентов, пришедших органическим путём, т.к. на их привлечение не было затрат

prof= profiles[profiles['channel']!='organic']
```

In [69]:
```
mean_cost_all = prof['acquisition_cost'].mean()
print('Средний CAC на одного пользователя для всего проекта:', round(mean_cost_all,2))
```

```
Средний CAC на одного пользователя для всего проекта: 1.13
```

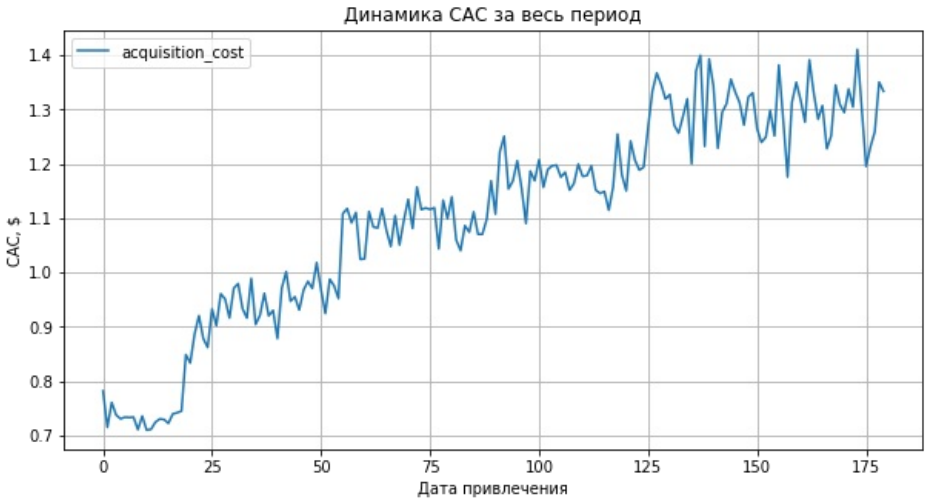
Теперь рассчитаем средний для каждого источника трафика:

Out[70]:

	channel	acquisition_cost
0	tiptop	2.799003
1	faceboom	1.113286
2	adnonsense	1.008054
3	lambdamediaads	0.724802
4	wahoonetbanner	0.602245
5	rocketsuperads	0.412095
6	opplecreativemedia	0.250000
7	yrabbit	0.218975
8	mediatornado	0.218717
9	leapbob	0.210172

Дополнительно посмотрим, как менаялась средние расходы на рекламу за весь период:

Out[71]: `Text(0.5, 1.0, 'Динамика CAC за весь период')`



А также рассмотрим, как менялся CAC за весь период по каналам привлечения:

Промежуточные выводы

Средний CAC на одного пользователя для всего проекта: 1.13 $. По каналам привлечения эта величина меняется от 2,8 (tiptop) до 0.21 (leapbob). За весь исследуемый период средняя стоимость привлечения клиента выросла примерно в 2 раза с 0,7 до 1,4. Рост среднего CAC всего проекта связан с ростом стоимости по одному каналу - tiptop (с 1 до 3,5), по остальным каналам средние CAC колебались примерно на одном уровне.

Наиболее дорогостоящим каналом привлечения пользователей является TipTop, однако количество и доля платящих пользователей, приходящих с этого источника, не так высоки. Это можно объяснить молодой аудиторией TipTop'а и, соответственно, не очень высокой их платежеспособностью.

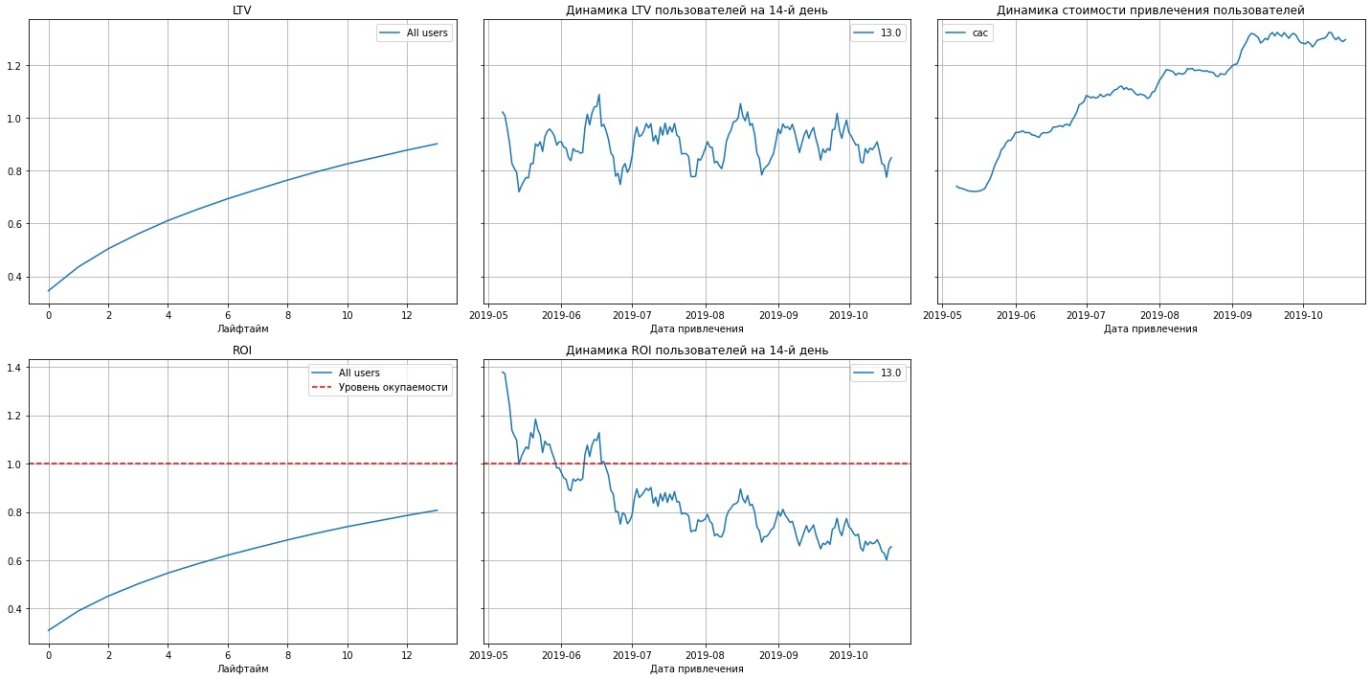
Шаг 5. Оценка окупаемости рекламы для привлечения пользователей

Анализ общей окупаемости рекламы

Рассчитаем и визуализируем LTV и ROI, вызвав функции get_ltv() и plot_ltv_roi():

Установим момент и горизонт анализа данных. По условию на календаре 1 ноября 2019 года, а в бизнес-плане заложено, что

пользователи должны окупаться не позднее чем через две недели после привлечения. В анализ не включаем органических пользователей, т.к. на них не было расходов на привлечение и их включение будет искажать итоговый результат анализа.

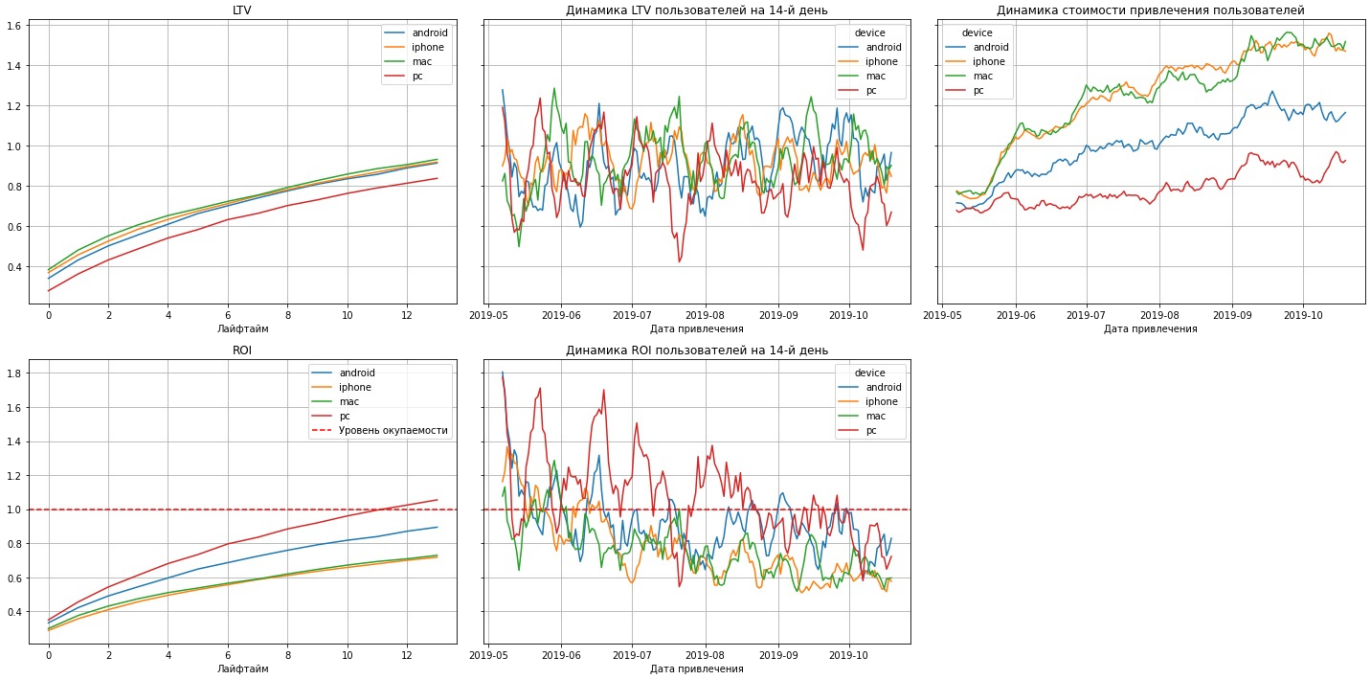


Выводы по графикам:

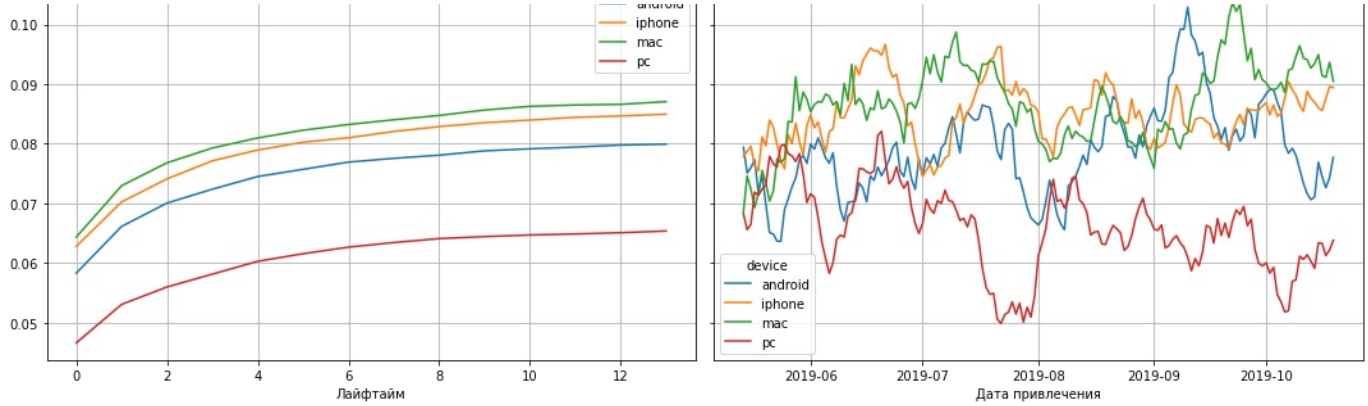
- вложенные в рекламу средства не окупаются, и на 14 день расходы на рекламу возвращаются в размере примерно 82%.
- стоимость привлечения пользователей за весь с начала мая по конец октября 2019 стабильно растёт
- LTV на 14 день подвержен колебаниям, но всё же относительно стабилен, чего не скажешь о возврате инвестиций, которые на 14 день неуклонно падают.

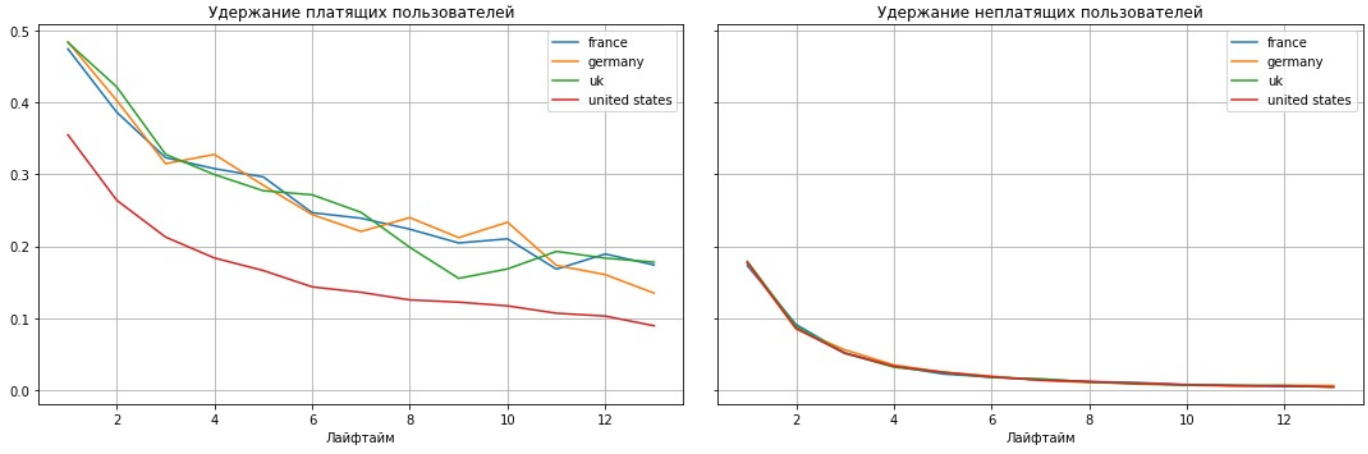
Наблюдаем, что динамика ROI за лайфтайм падает. При относительно стабильной динамике LTV, динамика CAC растёт с мая по конец октября. Эту закономерность мы наблюдаем в динамике ROI, что при сильном увеличении CAC, в равной степени падает динамика ROI пользователей.

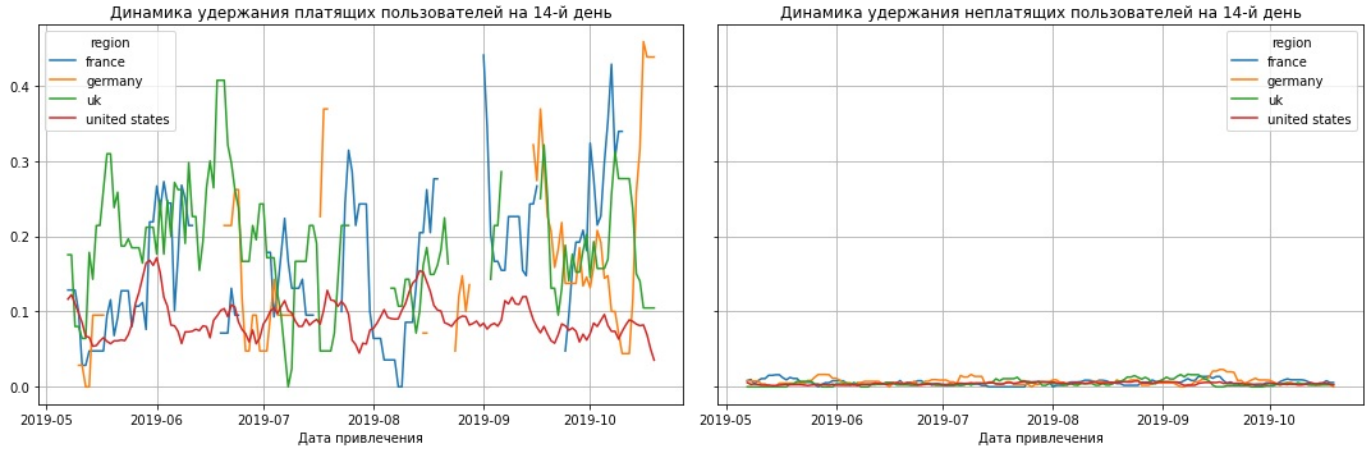
Анализ окупаемости рекламы с разбивкой по устройствам



Рассмотрим графики конверсии и удержания с разбивкой по устройствам:







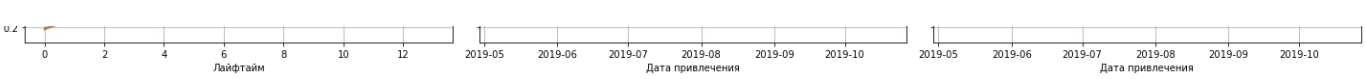
Выводы по графикам:

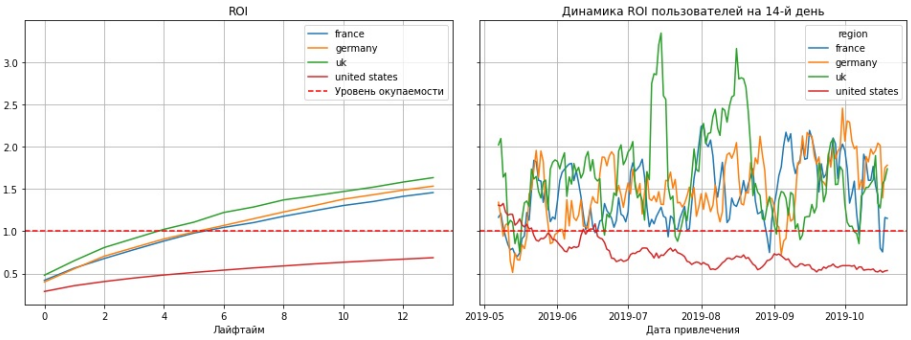
LTV для пользователей iphone, android, mac примерно одинакова и чуть ниже для пользователей pc, но при этом только для пользователей pc затраты на рекламу окупаются и являются самыми низкими по стоимости привлечения на одного клиента.

При этом видим интересную картину на графиках конверсии и удержания: в обоих случаях пользователи pc значительно отстают от пользователей iphone, android, mac.

В целом, с окупаемостью проблемы по всем устройствам кроме PC. Это значит, что у нас, по крайней мере, нет технических проблем, влияющих на монетизацию.

Анализ окупаемости рекламы с разбивкой по странам

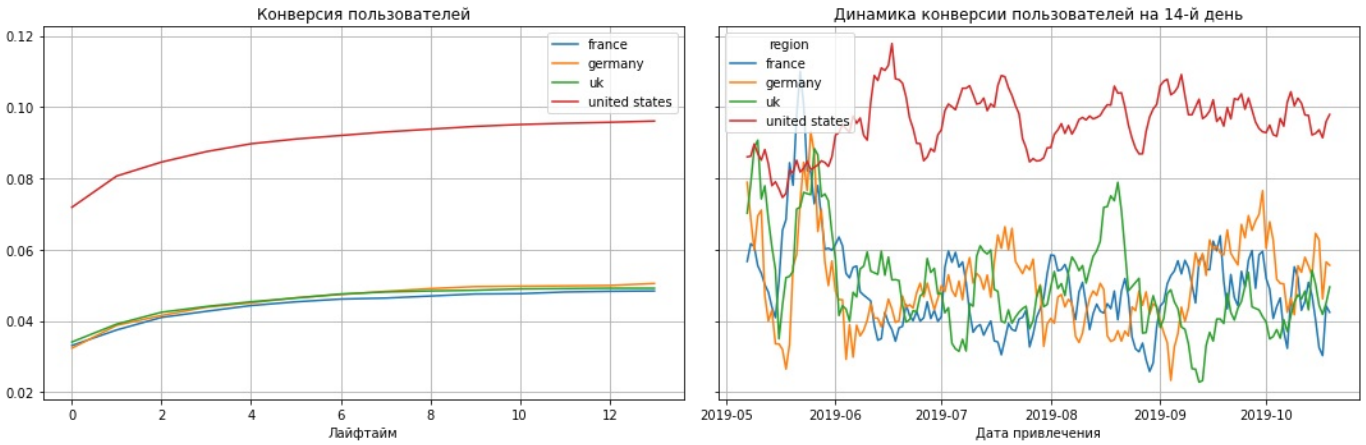


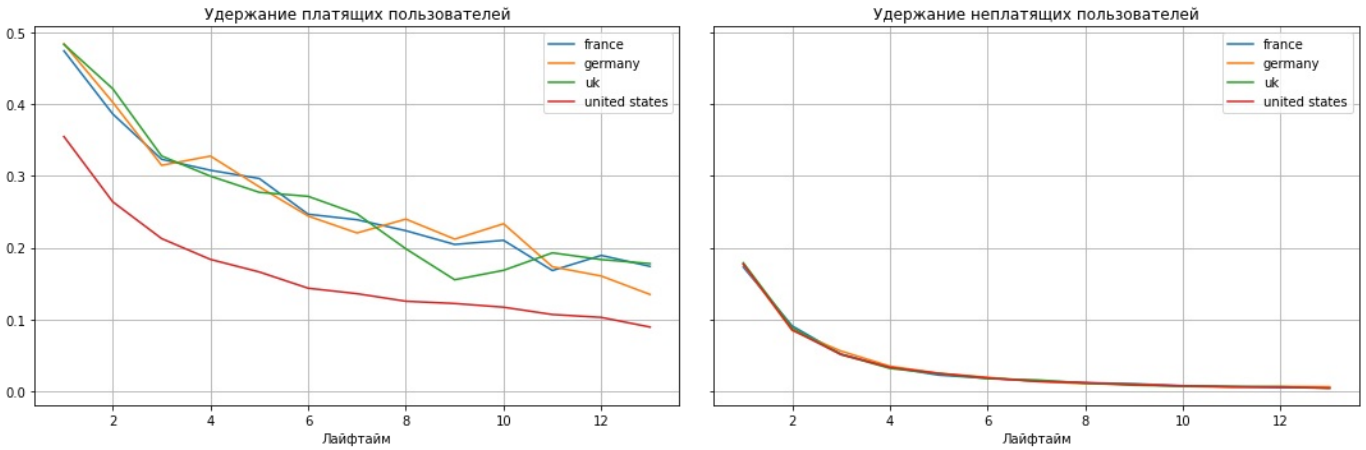


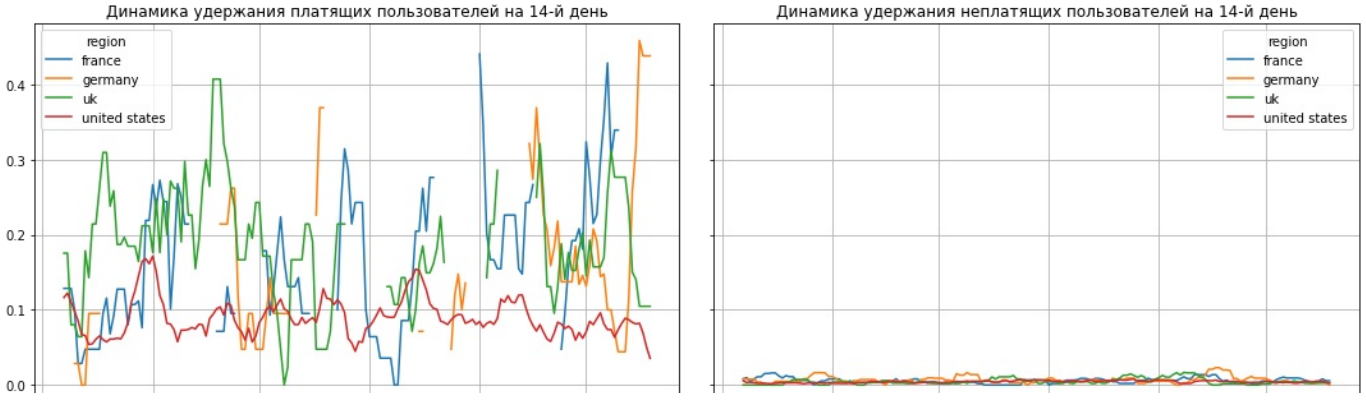
Выводы по графикам:

LTV пользователей США стабильно выше других стран на 0,2-0,3, но при этом привлечение таких клиентов неуклонно росло за весь период и совершенно не окупалось (около 70%). По остальным странам LTV ниже и примерно одинаковы между странами (лидирует среди них Великобритания), но при этом намного ниже и стомость привлечения пользователей. К тому же, по сравненению с США, по остальным странам CAC снизилась в начале июня 2019 и оставалась на одном уровне в течение всего периода и затраты на рекламу по Франции, Германии, Великобритании окупаются на 4-6 день после привлечения клиента.

Посмотрим детальнее в чём проблема с клиенами из США, для этого рассмотрим графики конверсии и удержания с разбивкой по странам:







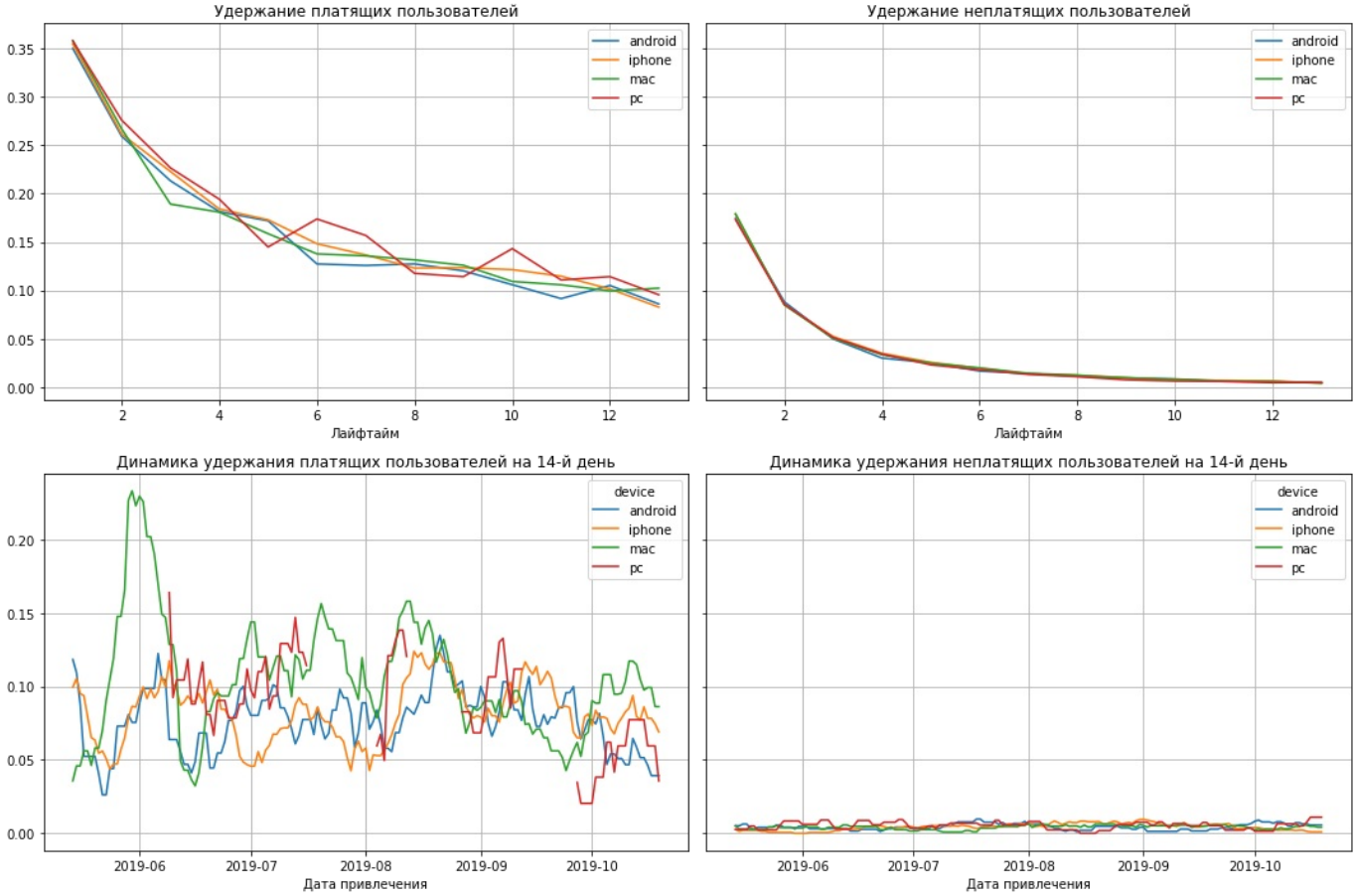
Выводы к графикам:

по графикам видим, что конверсия пользователей в США самая высокая относительно других стран, но вот с удержанием платящих пользователей есть большие проблемы, этот показатель сильно отстает от других стран.

Провирим, может ли проблемы с удержанием быть связаны с используемыми устройствами. Отфильтруем пользователей из США:

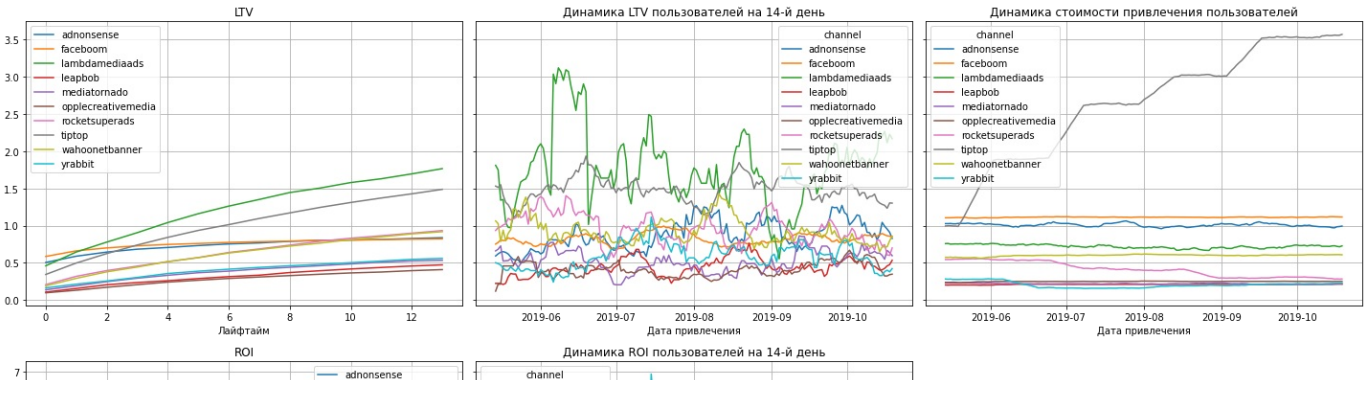
Out[80]:

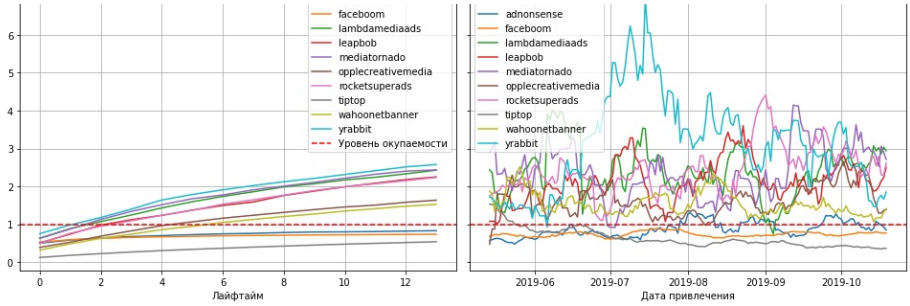
	user_id	first_ts	channel	device	region	dt	month	payer	acquisition_cost
0	599326	2019-05-07 20:58:57	faceboom	mac	united states	2019-05-07	2019-05-01	True	1.088172
1	4919697	2019-07-09 12:46:07	faceboom	iphone	united states	2019-07-09	2019-07-01	False	1.107237
4	31989216	2019-10-02 00:07:44	yrabbit	iphone	united states	2019-10-02	2019-10-01	False	0.230769



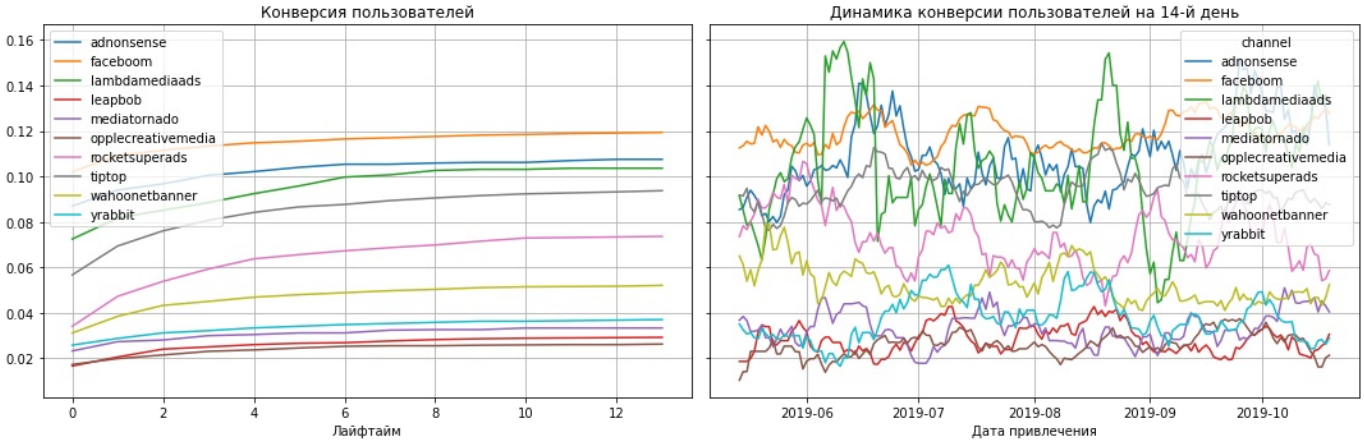
По графикам видим, что какой-то определённый тип устройства (или неполадки с ним) на удержение клиентов из США критического влияния не имеет, но по более значительным колебаниям удержания в положительную сторону клиентов pc можно предположить, что на остальных устройствах есть какие-то примерно одинаковые минусы, на которые необходимо обратить внимание.

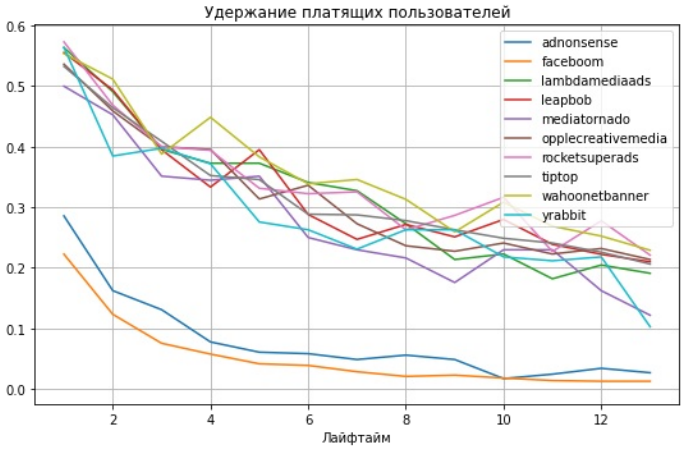
Анализ окупаемости рекламы с разбивкой по рекламным каналам

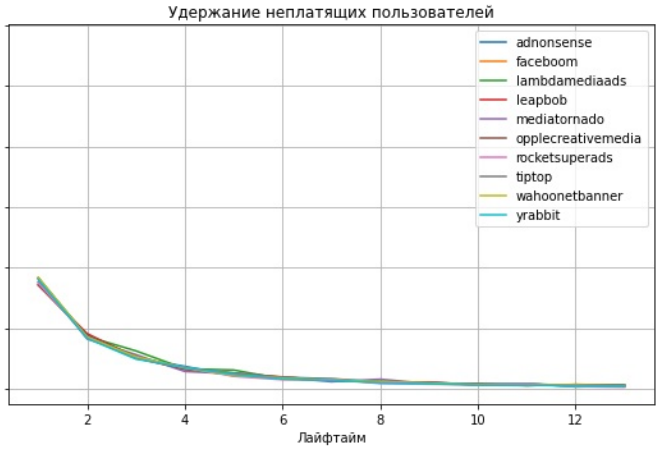




Рассмотрим графики конверсии и удержания с разбивкой по рекламным каналам:







Выводы к графикам:

К каналам, по которым инвестиции не возвращаются в полном объеме относятся: adnonsense, faceboom, tiptop. К тому же по каналу tiptop стоимость привлечения пользователей постоянно росла, по другим каналам этот показатель оставался стабильным на протижении всего периода. Видим также, что по каналам adnonsense, faceboom - самое низкое удержание платящих пользователей.

Шаг 6. Выводы и рекомендации

Причины неэффективности привлечения пользователей

1) Дорогостоящая и постоянно растущая в цене реклама в канале tiptop, что косвенно можем наблюдать по окупаемости рекламы на различных устройствах - окупается реклама только у пользователей pc, т.к. для этих пользователей канал tiptop не досупен. Также убыточными каналами привлечения являются adnonsense, faceboom, клиенты из этих каналов хуже всего удерживаются.

2) Низкое удержание платящих пользователей из США.

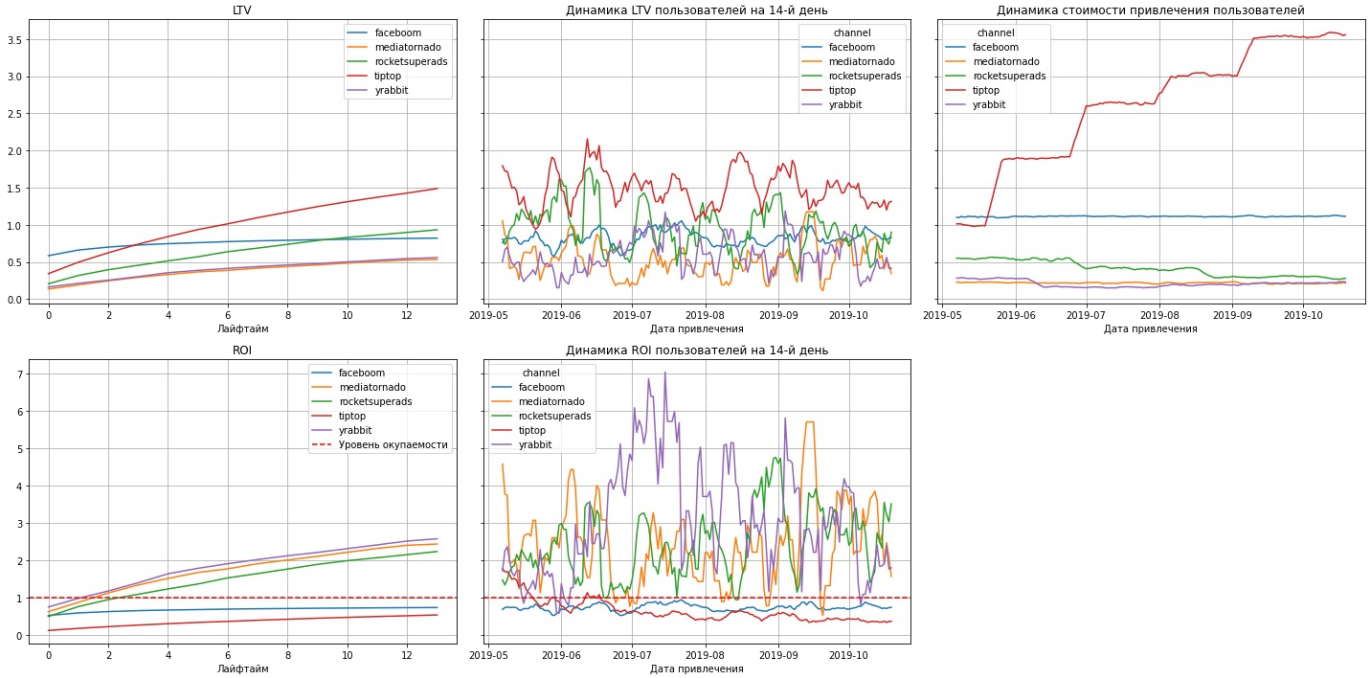
Рекомендации для отдела маркетинга для повышения эффективности

1) Необходимо отказаться от рекламы в каналах tiptop, adnonsense, faceboom и перераспределить рекламный бюджет в пользу перспективных окупаемых каналов: yrabbit, mediatornado,lambdamediaads, leapbob, rocketsuperads, по которым вложенные средства возвращаются в среднем на второй день привлечения клиента.

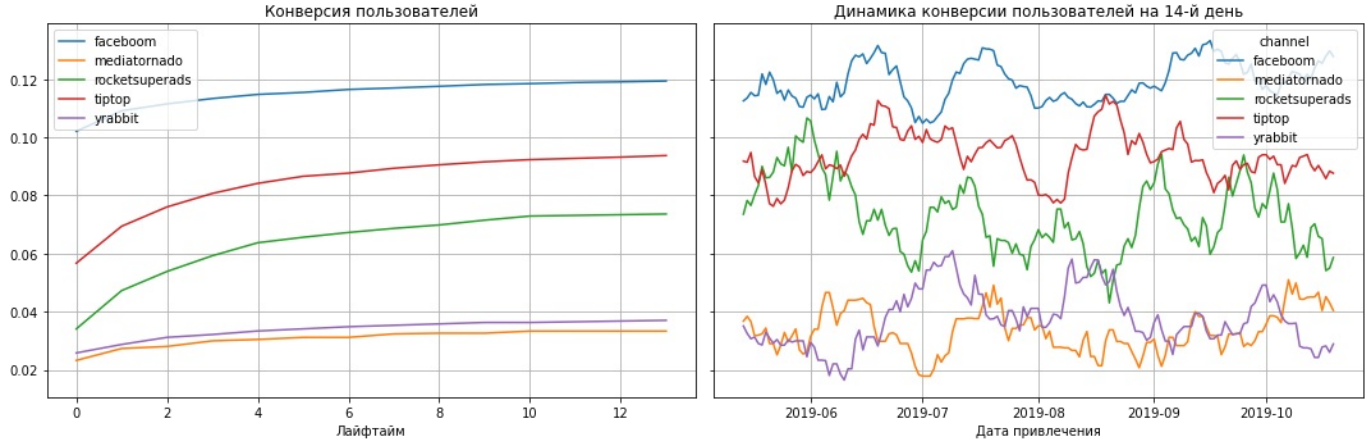
2) Необходимо провести маркетинговое исследование среди пользователей США, что о причинах их неудовлетворённости при использовании сервиса, в том числе дополнительно посмотреть является ли использование сервиса на pc чем-то более удобным по сравнению с другими устройствам.

Рынок США и работа каналов привлечения

In [85]:
```
profiles_USA = prof[prof['region']=='united states']
```

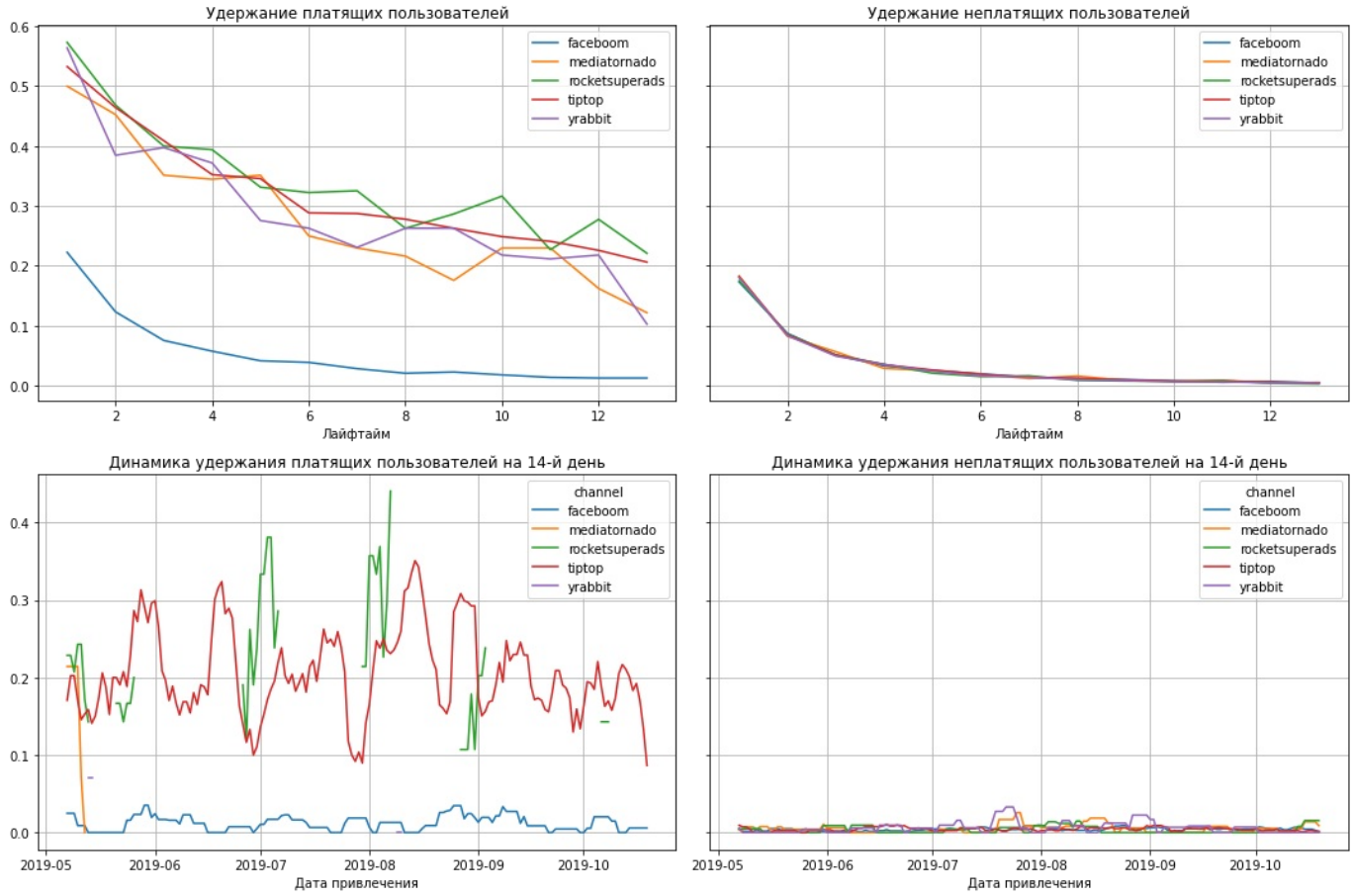


Рассмотрим графики конверсии и удержания с разбивкой по каналам привлечения по рынку США:



Конверсия пользователей в США с разбивкой по каналам привлечения:

по графика видим, что на рынке США представлены не все каналы, а только faceboom, mediatornado, rocketsuperads,tiptop, yrabbit. Лучшим по конверсии является faceboom, на втором месте tiptop и затем rocketsuperads.

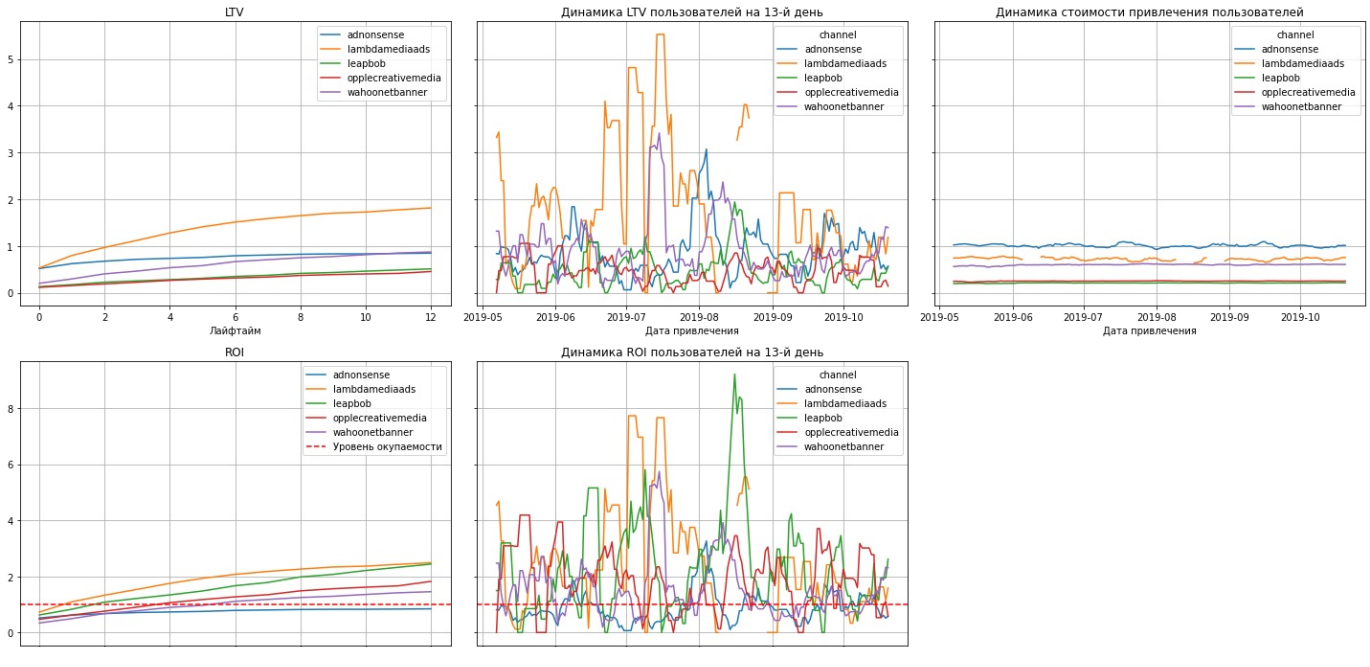


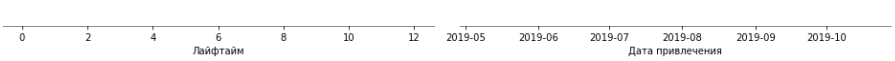
А вот по удержанию faceboom находится на последнем месте, тогда как остальные каналы показывают примерно одинаковые параметры по удержанию. По графику "Динамика удержания платящих пользователей на 14-й день" видим, что лучшие показатели по удержанию на 14-й день привлечения у каналов rocketsuperads,tiptop.

Рекомендация для маректологов: необходимо проверить, с чем связано такое низкое удержание платящих пользователей, привлеченных через канал faceboom, который в результате является убыточным по вложенным средствам, и отказаться от канала tiptop, который по окупаемости инвестиций является убыточным.

Рынок Великобритании и работа каналов привлечения

In [89]:
```
profiles_uk = prof[prof['region']=='uk']
```

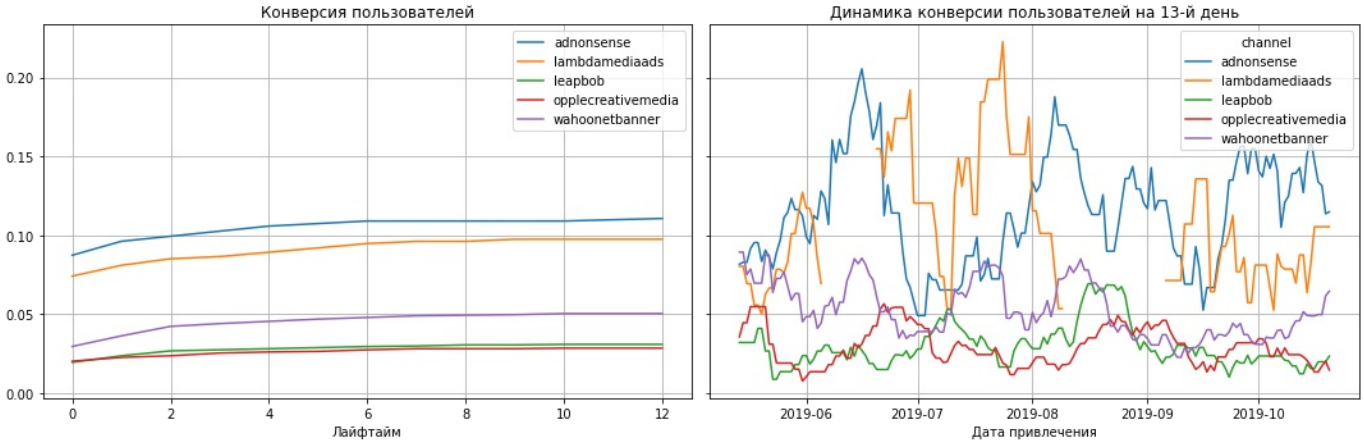


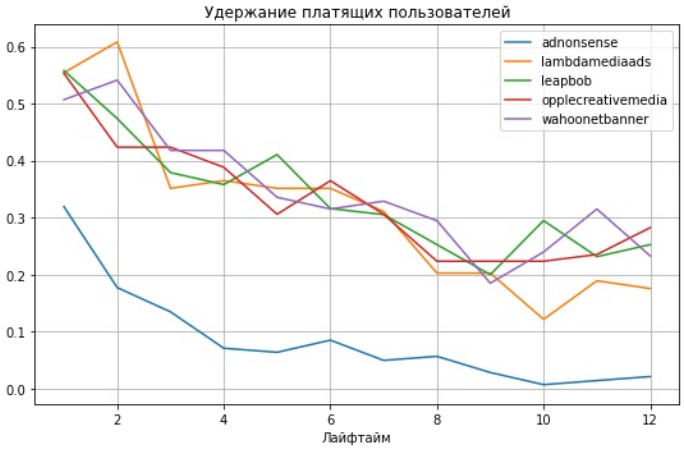


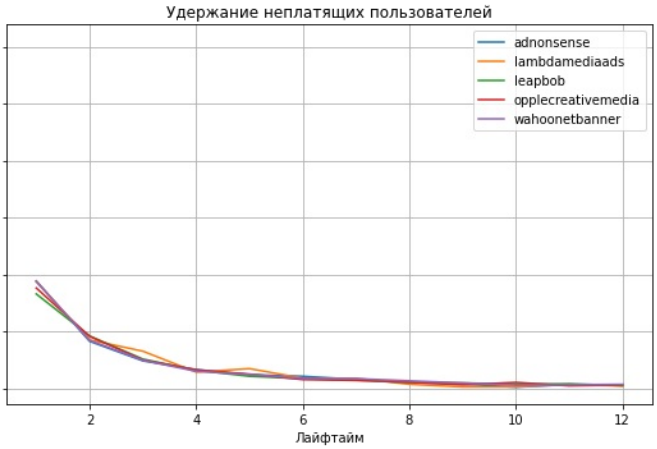
На рынке Великобритании пользователи привлекались из каналов:

- adnonsense
- opplecreativemedia
- lambdamediaads
- leapbob
- wahoonetbanner Из них не окупался только один канал: adnonsense

Рассмотрим графики конверсии и удержания с разбивкой по каналам по рынку Великобритании:







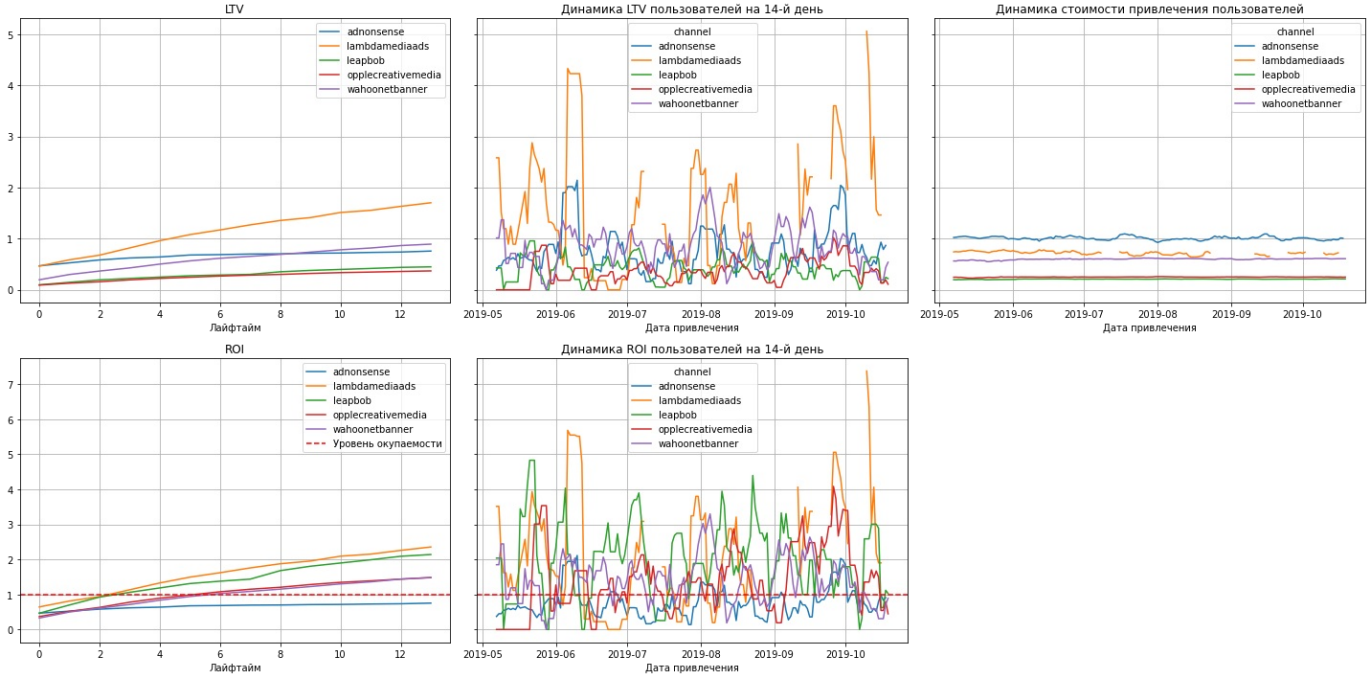
Хотя по конверсии пользователей канал adnonsense лидирует, но он самый последний по удержанию платящих пользователей, что и сказалось на его окупаемости.

Рекомендации для маректологов: 1) для рынка Великобритании рассмотреть возможность использования канала faceboom

2) проверить, с чем связано низкое удержание платящих пользователей из канала adnonsense

Рынок Франции и работа каналов привлечения

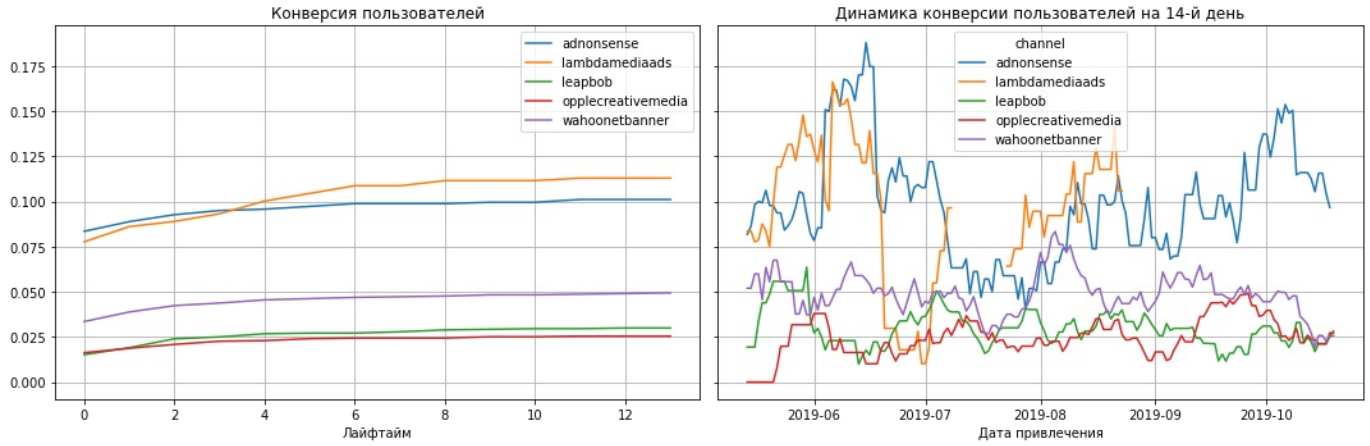
In [93]:
```
profiles_france = prof[prof['region']=='france']
```



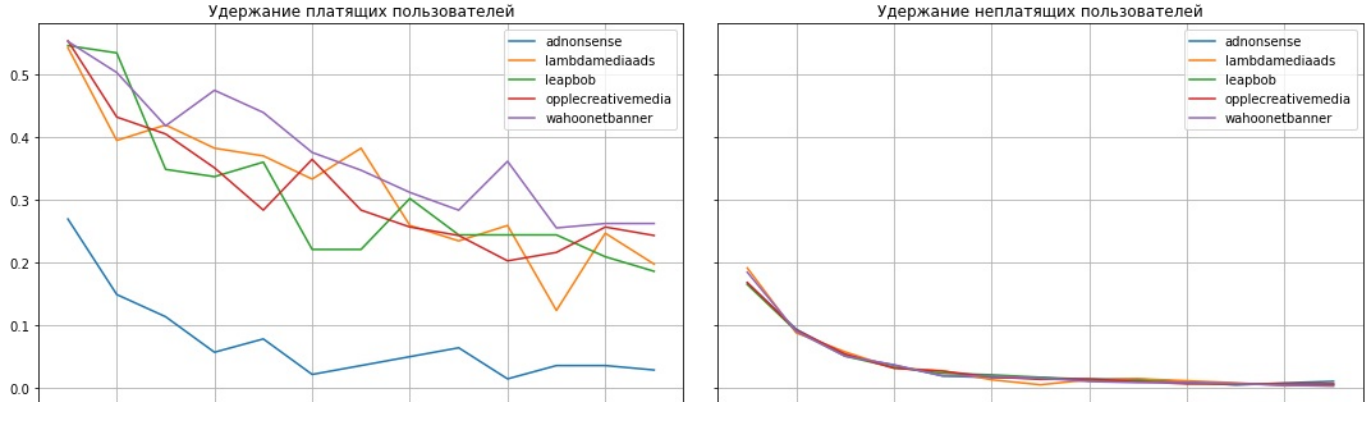
На рынке Франции использовались каналы:

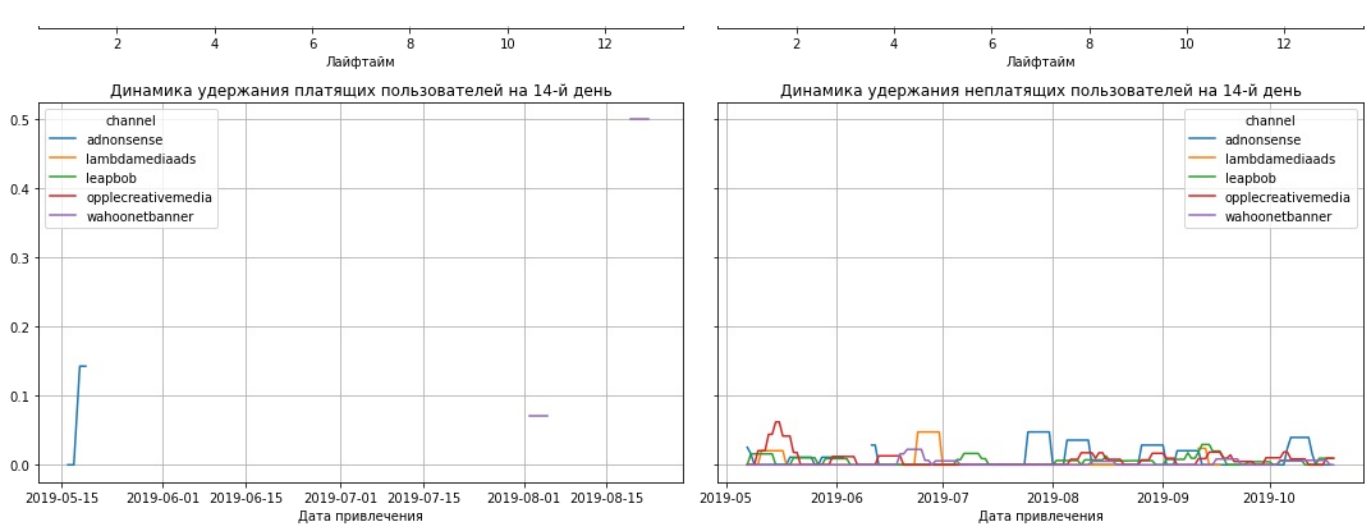
- adnonsense
- lambdamediaads
- leapbob
- opplecreativemedia
- wahoonetbanner

Аналогично рынку Великобритании канал adnonsense является убыточным. По окупаемости лидирует канал lambdamediaads.



Видим, что по конверсии пользователей лидирует канал lambdamediaads, а вот последним по этому параметру является opplecreativemedia, и всё же он является окупаемым, в отличии от adnonsense, который на втором месте по конверсии.





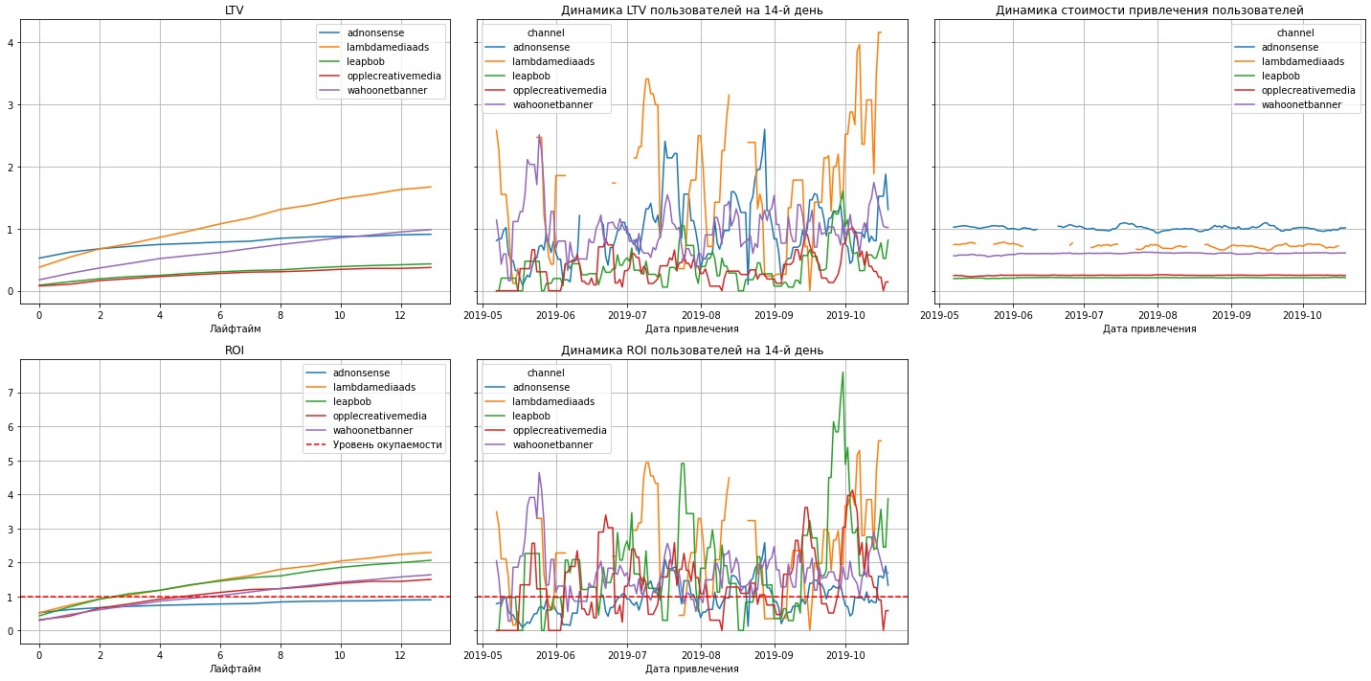
Аналогично рынку Великобритании, на рынке Франции канал adnonsense последний по удержанию пользователей. Лидирующим по удержанию является wahoonetbanner.

Рекомендации для маркетологов: 1) увеличить вложения рекламу на канале lambdamediaads, т.к. он самый окупаемый по вложениям

2) выяснить причину низкого удержания по каналу adnonsense, до выяснения причин, отказаться от его испльзования, как от убыточного.

Рынок Германии и работа каналов привлечения

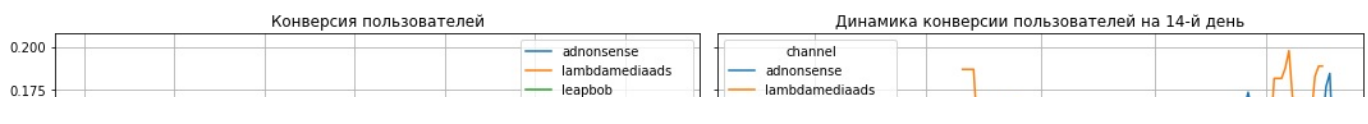
In [97]:
```
profiles_germany = prof[prof['region']=='germany']
```

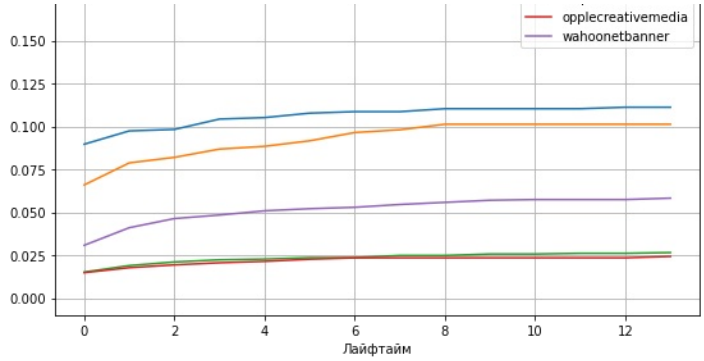


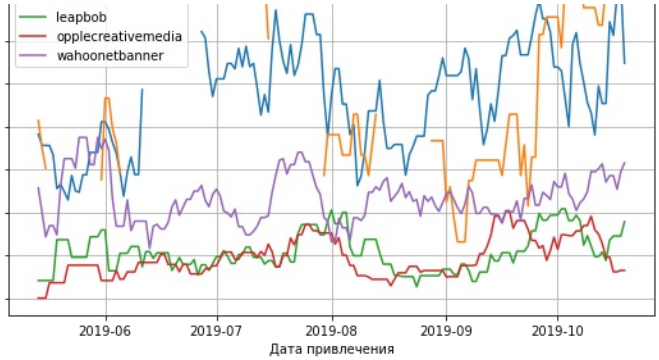
На рынке Германии испльзовались каналы:

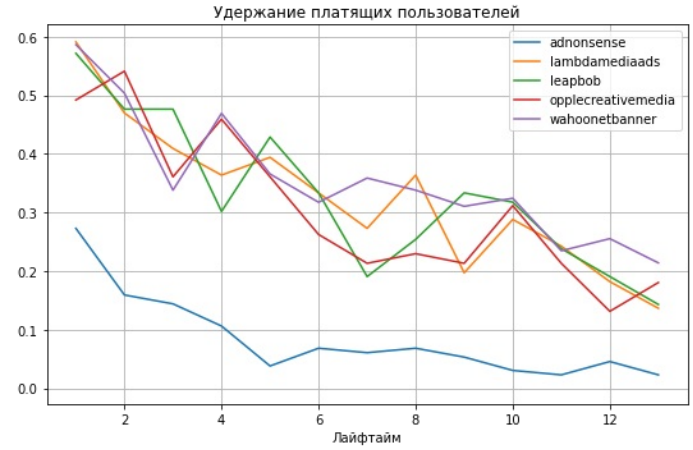
- adnonsense
- lambdamediaads
- leapbob
- opplecreativemedia
- wahoonetbanner

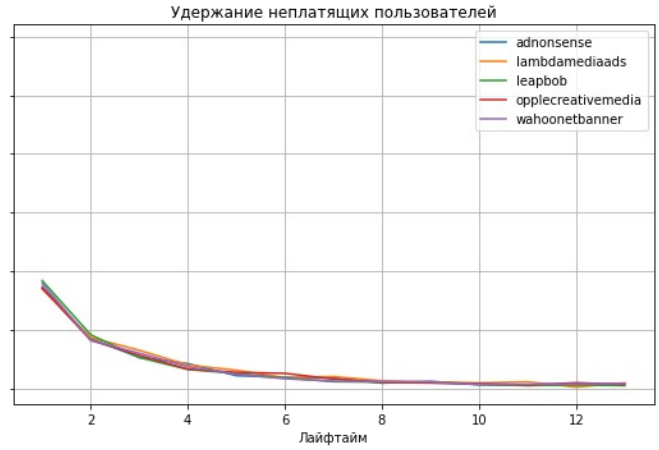
Аналогично рынку Великобритании и Франции - канал adnonsense является убыточным, по окупаемости лидирует канал lambdamediaads.











Видим, что по конверсии пользователей лидирует канал adnonsense, на втором месте lambdamediaads, а вот последним по этому параметру является opplecreativemedia, и всё же он является окупаемым, в отличии от adnonsense.

Аналогично рынку Великобритании и Франции, в Германии канал adnonsense последний по удержанию пользователей. Остальные каналы держаться примерно в одном диапазоне.

Рекомендации для маркетологов: 1) увеличить вложения рекламу на канале lambdamediaads, т.к. он самый окупаемый по вложениям

2) выяснить причину низкого удержания по каналу adnonsense, до выяснения причин, отказаться от его испльзования, как от убыточного.

Видим, что по всем европейским рынкам есть проблема с удержанием клентов из канала adnonsense, который является самым лучшим по показателю конверсии. Есть отличия в испльзовании рекламных калалов в США и в Европе. Те, что испльзуются в США, не испльзуются в Европе и наоборот. Как минимум faceboom стоит рассмотреть для испльзования в Европе, возможно также рассмотреть испльзование в Европе - прибыльных в США каналов rocketsuperads, mediatornado, rabbit, если они актуальны для европейских стран.

А для США рассмотреть испльзование самых прибыльных для Европы каналов - lambdamediaads, leapbob, если эти каналы актуальны для США.