

Table of Contents

- 1 Введение
- 2 Извлечение и подготовка данных
 - 2.1 Удаление дубликатов
 - 2.2 Вывод по разделу
- 3 Изучение данных
 - 3.1 Диапазон дат
 - 3.2 Уникальные пользователи
 - 3.3 Выводы по разделу
- 4 Воронка событий
 - 4.1 Типы событий
 - 4.2 Воронка продаж
 - 4.3 Выводы по разделу
- 5 Изучение результата эксперимента
 - 5.1 Группы
 - 5.2 Самое популярное событие
 - 5.3 Проверка контрольных групп
 - 5.4 Группа с изменённым шрифтом
 - 5.4.1 Сравнение тестовой группы с контрольными
 - 5.4.2 Сравнение тестовой группы с объединёнными контрольными
 - 5.5 Поправка на множественную проверку гипотез
- 6 Заключение и выводы

Введение

В работе будет рассмотрено приложение для продажи продуктов питания.

Дизайнеры сделали новые шрифты, продвигают идею об их замене во всём приложении.

У менеджеров есть опасения, что это повлияет на поведение пользователей и негативно отразится на выручке.

Перед внедрением нового шрифта решено провести A/A/B-тест.

Пользователей разбили на 3 группы: 2 контрольные со старыми шрифтами и одну экспериментальную — с новыми.

Далее будет проанализировано поведение пользователей мобильного приложения.

Проведём проверку воронки продаж.

Исследуем результаты A/A/B-теста, выясним, влияет ли изменение шрифтов на поведение пользователей.

```
In [1]: import os
import pandas as pd
import numpy as np
import math as mth
from scipy import stats as st
```

```
import plotly.express as px
import plotly.graph_objects as go
```

```
In [2]: def dates(date_col):
        """ функция напечатает минимальную и максимальную дату,
            а также период наблюдений в днях.
            date_col: Series с датами.
        """
        min_date = date_col.min()
        max_date = date_col.max()
        # к разнице дат добавим 2, чтобы включить в период 1 и последнюю даты
        period = pd.Timedelta(max_date - min_date).days + 1
        print(f'Начальная дата: {min_date}.')
        print(f'Конечная дата: {max_date}.')
        print(f'Период наблюдений: {period} дней.')
```

```
In [3]: def difference(a, b):
        """Функция вернёт разницу между величинами a и b
        """
        diff = abs(a - b)
        diff_percent = round(diff * 100 / min(a, b), 1)
        return diff, diff_percent
```

```
In [4]: def check_group_diff(target_1, target_2, base_1, base_2, alpha=.05):
        """Функция проверит гипотезу о равенстве долей при помощи z-теста.
            группа 1: target_1, base_1
            группа 2: target_2, base_2
            α по умолчанию 0.05
        """
        alpha = alpha # критический уровень статистической значимости

        # пропорция успехов в первой группе:
        p1 = target_1/base_1

        # пропорция успехов во второй группе:
        p2 = target_2/base_2

        # пропорция успехов в комбинированном датасете:
        p_combined = (target_1 + target_2) / (base_1 + base_2)

        # разница пропорций в датасетах
        difference = p1 - p2

        # считаем статистику в ст.отклонениях стандартного нормального распределения
        z_value = difference / mth.sqrt(
            p_combined * (1 - p_combined) * (1 / base_1 + 1 / base_2)
        )

        # задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
        distr = st.norm(0, 1)

        p_value = (1 - distr.cdf(abs(z_value))) * 2

        print('p-значение: ', p_value)

        if p_value < alpha:
            print('Отвергаем нулевую гипотезу: между долями есть значимая разница')
        else:
            print(
                'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разные'
            )
```

```
In [5]: def prob_check(alpha, k):
        """ функция посчитает вероятность получения ложно положительного результата
```

```

k - число сравнений
alpha - уровень статистической значимости
"""
false_positive_prob = 1-(1-alpha)**k
return false_positive_prob

```

Извлечение и подготовка данных

```

In [6]: pth1 = '/datasets/logs_exp.csv'
pth2 = r"C:\Users\Vladislav Sinelnikov\OneDrive\Documents\YaDA\module_2\logs_exp.csv"

if os.path.exists(pth1):
    data = pd.read_csv(pth1, sep='\\t', engine='python')
elif os.path.exists(pth2):
    data = pd.read_csv(pth2, sep='\\t', engine='python')
else:
    print('Something is wrong')

```

```

In [7]: data.info()
data.head(3)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244126 entries, 0 to 244125
Data columns (total 4 columns):
EventName          244126 non-null object
DeviceIDHash        244126 non-null int64
EventTimestamp      244126 non-null int64
ExpId               244126 non-null int64
dtypes: int64(3), object(1)
memory usage: 7.5+ MB

```

```

Out[7]:

```

	EventName	DeviceIDHash	EventTimestamp	ExpId
0	MainScreenAppear	4575588528974610257	1564029816	246
1	MainScreenAppear	7416695313311560658	1564053102	246
2	PaymentScreenSuccessful	3518123091307005509	1564054127	248

Пропуски отсутствуют. Требуется изменить тип данных для 'EventTimestamp'.

```

In [8]: # переименуем колонки
data = (
    data.rename(columns={
        'EventName': 'event_name',
        'DeviceIDHash': 'device_id_hash',
        'EventTimestamp': 'event_timestamp',
        'ExpId': 'exp_id'
    })
)

```

```

In [9]: # изменение типа данных для 'event_timestamp'
data['event_timestamp'] = pd.to_datetime(data['event_timestamp'], unit='s')

```

```

In [10]: # добавление отдельных колонок с датой и временем
data['date'] = pd.to_datetime(data['event_timestamp']).dt.date
data['time'] = pd.to_datetime(data['event_timestamp']).dt.time

```

Удаление дубликатов

```

In [11]: print(f'В данных {data[data.duplicated() == True].shape[0]} дублирующих записей.\n')

```

```
f'Это {data[data.duplicated() == True].shape[0] / data.shape[0] * 100:.1f}% от всех
```

В данных 413 дублирующих записей.
Это 0.2% от всех записей, можем их удалить.

```
In [12]: # удаление явных дублей
data = data.drop_duplicates()
```

Вывод по разделу

Нам предоставлен довольно качественный датасет без пропусков и малым количеством дублей (0.2%).

Дубли удалены, перейдём к изучению данных.

Изучение данных

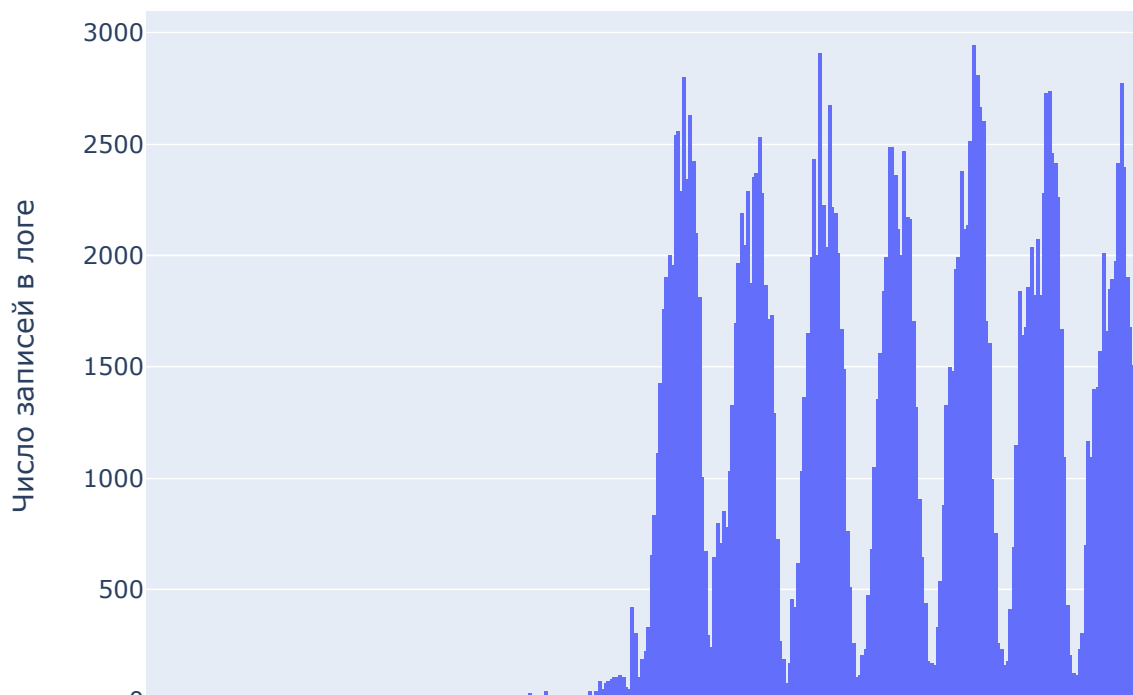
Диапазон дат

```
In [13]: dates(data['date'])
```

Начальная дата: 2019-07-25.
Конечная дата: 2019-08-07.
Период наблюдений: 14 дней.

```
In [14]: # гистограмма по датам
fig = px.histogram(data,
                    x='event_timestamp',
                    nbins=500,
                    title='Гистограмма использования приложения по датам')
fig.update_yaxes(title_text='Число записей в логе')
fig.update_xaxes(title_text='Дата')
fig.show()
```

Гистограмма использования приложения по датам



Дата

Выводы из графика

- Посещение приложение циклично с периодом в сутки.
- Тест начался 31 июля в 22:00, можно отдельно выделить участок 31 июля - 1 августа на графике.
- В 00:00 - 01:00 самая низкая активность. Пользователи начинают интересоваться едой уже с 03:00.

По гистограмме видно, что число суточных событий до 31 июля включительно значительно меньше числа суточных событий после.

Оставим для изучения события, начиная с 1 августа включительно.

```
In [15]: # число записей до 1 августа
before_augst = data[data['event_timestamp'] < '2019-08-01 00:00:00'].shape[0]
print(f'Из рассмотрения убираем {before_augst} (или {before_augst / data.shape[0] * 100
```

Из рассмотрения убираем 2826 (или 1.2%) записей.

```
In [16]: data_upd = data[data['event_timestamp'] >= '2019-08-01 00:00:00']
```

```
In [17]: # даты в оставшемся датасете
dates(data_upd['date'])
```

Начальная дата: 2019-08-01.

Конечная дата: 2019-08-07.

Период наблюдений: 7 дней.

Уникальные пользователи

```
In [18]: # число уникальных пользователей
unique_users = data_upd['device_id_hash'].nunique()
print(f'Уникальных пользователей за период: {unique_users}.')
```

Уникальных пользователей за период: 7534.

Проверим, уникальны ли пользователи в каждой группе.

```
In [19]: # проверка уникальности пользователей по каждой группе
users_246 = set(data_upd.query('exp_id == 246')['device_id_hash'])
users_247 = set(data_upd.query('exp_id == 247')['device_id_hash'])
users_248 = set(data_upd.query('exp_id == 248')['device_id_hash'])
print('Результат проверки:')
if (not users_246 & users_247
    and not users_247 & users_248
    and not users_246 & users_248):
    print('В каждой группе только уникальные пользователи.')
else:
    print('В одной из групп есть пересечения, требуются дополнительные проверки.')
```

Результат проверки:

В каждой группе только уникальные пользователи.

```
In [20]: # число пользователей по группам
(
    data_upd.groupby('exp_id', as_index=False)
    .agg({'device_id_hash': 'nunique'})
```

```
.rename(columns={'device_id_hash': 'unique_users'})
)
```

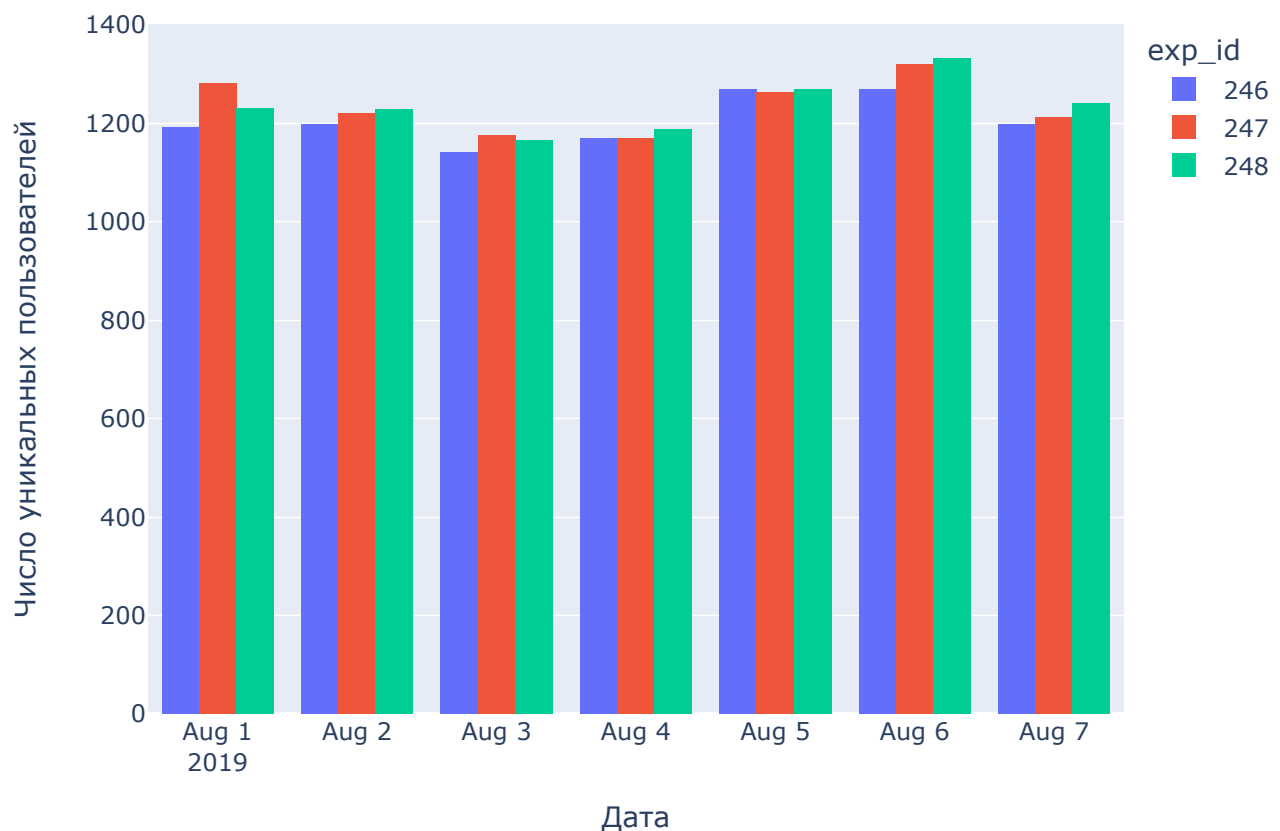
Out[20]:

	exp_id	unique_users
0	246	2484
1	247	2513
2	248	2537

```
In [21]: # число уникальных пользователей по датам в каждой группе
chaeck_groups = (
    data_upd.groupby(['date', 'exp_id'], as_index=False)
    .agg({'device_id_hash': 'nunique'})
    .sort_values(by='date', ascending=True)
    .rename(columns={'device_id_hash': 'unique_devices'})
)
fig = px.histogram(chaeck_groups,
                    x='date',
                    y='unique_devices',
                    color='exp_id',
                    barmode='group',
                    nbins=10,
                    title='Гистограмма числа уникальных пользователей по дням')
fig.update_yaxes(title_text='Число уникальных пользователей')
fig.update_xaxes(title_text='Дата')

fig.show()
```

Гистограмма числа уникальных пользователей по дням



In [22]: # число событий на пользователя за весь период

```

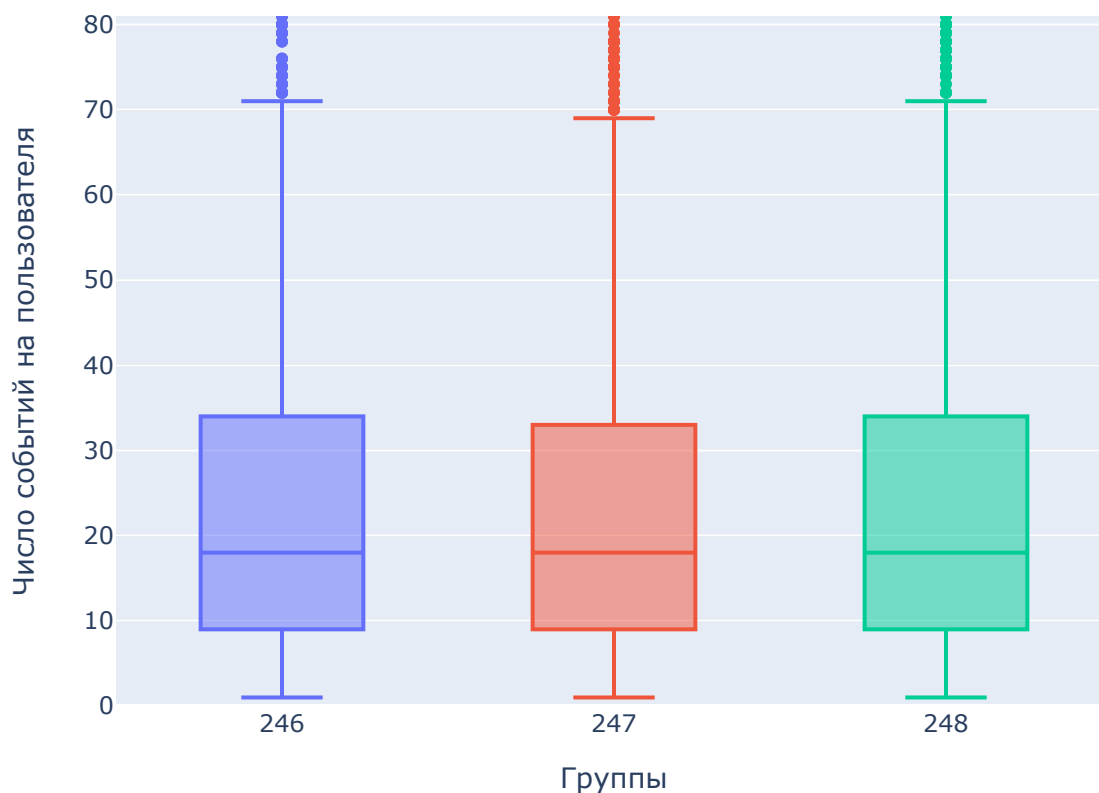
unique_users_activity = (
    data_upd.groupby(['device_id_hash', 'exp_id'], as_index=False)
    .agg({
        'event_timestamp': 'nunique',
        'event_name': 'nunique'
    })
    .rename(columns={
        'event_timestamp': 'events',
        'event_name': 'event_types_total'
    })
    .sort_values(by=['events'], ascending=False)
)

fig = go.Figure()

for group in unique_users_activity['exp_id'].sort_values().unique():
    fig.add_trace(
        go.Box(
            y=unique_users_activity.query('exp_id == @group')['events'],
            name=group.astype(str),
            showlegend=False))
fig.update_layout(title = 'Диаграммы размаха числа событий на пользователя по группам')
fig.update_yaxes(
    title_text='Число событий на пользователя',
    range=[0, np.percentile(unique_users_activity['events'], 95)])
fig.update_xaxes(title_text='Группы')
fig.show()

```

Диаграммы размаха числа событий на пользователя по группам



Выводы по разделу

- Для дальнейшей обработки данных в датасете оставлены записи, начиная от 1 августа и позже.
- В каждой группе примерно одинаковое число пользователей.
- В каждой группе только уникальные пользователи.

Воронка событий

Типы событий

```
In [23]: # типы событий
events_funnel = (
    data_upd.groupby(['event_name'], as_index=False)
    .agg({'event_timestamp': 'count', 'device_id_hash': 'nunique'})
    .rename(columns={'event_timestamp': 'event_count', 'device_id_hash': 'unique_users'})
    .sort_values(by='event_count', ascending=False)
)
events_funnel
```

```
Out[23]:
```

	event_name	event_count	unique_users
1	MainScreenAppear	117328	7419
2	OffersScreenAppear	46333	4593
0	CartScreenAppear	42303	3734
3	PaymentScreenSuccessful	33918	3539
4	Tutorial	1005	840

```
In [24]: ev_count = events_funnel['event_count'].sum()
print(f'В логе {ev_count} события.')
```

В логе 240887 события.

В логах всего 5 типов событий, 239 882 события.

Реже всего пользователи пользуются руководством.

Использование tutorиала не входит в цепочку покупки, исключим его из воронки.

Последовательность событий следующая:

1. Открытие главного экрана (MainScreenAppear).
2. Предложение (OffersScreenAppear).
3. Открытие корзины (CartScreenAppear).
4. Подтверждение успешной оплаты (PaymentScreenSuccessful).

```
In [25]: # исключим Tutorial из рассмотрения
events_funnel = events_funnel.query('event_name != "Tutorial"')
```

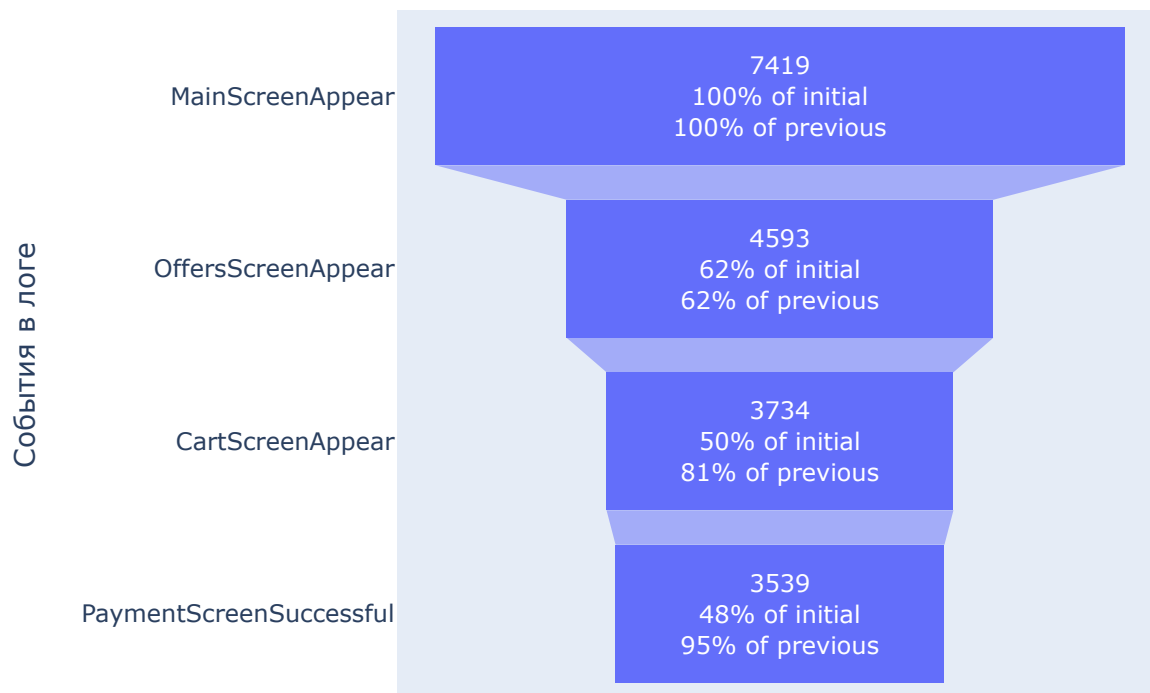
Воронка продаж

```
In [26]: # воронка по всем группам
fig = go.Figure()
fig.add_trace(go.Funnel(
    y=events_funnel['event_name'],
    x=events_funnel['unique_users'],
    textinfo = 'value + percent initial + percent previous'))
fig.update_layout(title='Воронка продаж')
```



```
fig.update_yaxes(title_text='События в логге')
fig.show()
```

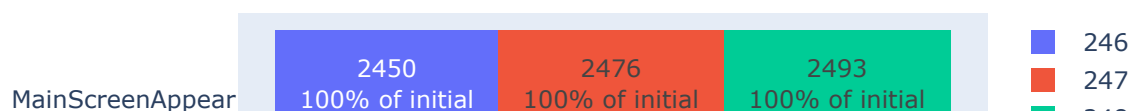
Воронка продаж

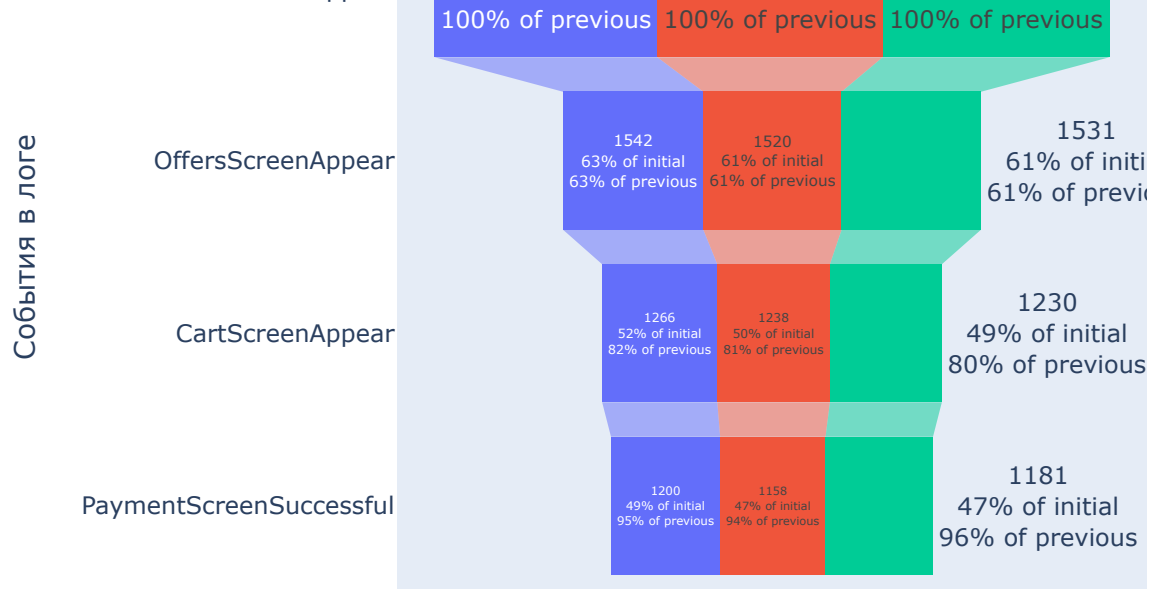


```
In [27]: # воронка с разбивкой по группам
events_funnel_groups = (
    data_upd.query('event_name != "Tutorial"')
    .groupby(['exp_id', 'event_name'], as_index=False)
    .agg({'event_timestamp': 'count', 'device_id_hash': 'nunique'})
    .rename(columns={'event_timestamp': 'event_count', 'device_id_hash': 'unique_users'})
    .sort_values(by='event_count', ascending=False)
)

fig = go.Figure()
for group in events_funnel_groups['exp_id'].sort_values().unique():
    fig.add_trace(go.Funnel(
        y=events_funnel_groups.query('exp_id == @group')['event_name'],
        x=events_funnel_groups.query('exp_id == @group')['unique_users'],
        name=group.astype(str),
        textinfo = 'value + percent initial + percent previous'))
fig.update_layout(title='Воронка продаж с разбиением по группам')
fig.update_yaxes(title_text='События в логге')
fig.show()
```

Воронка продаж с разбиением по группам





Выводы по разделу

- Определена последовательность событий и построена воронка.
- Больше всего пользователей теряется при переходе от главного экрана к экрану с предложением.
- Если пользователь сформировал корзину, скорее всего произойдёт и оплата.
- Tutorial не относится к цепочке продаж, исключили эти записи из рассмотрения.

Изучение результата эксперимента

Группы

```
In [28]: # число уникальных пользователей в каждой группе
unique_users_246 = data_upd.query('exp_id == 246')['device_id_hash'].nunique()
unique_users_247 = data_upd.query('exp_id == 247')['device_id_hash'].nunique()
unique_users_248 = data_upd.query('exp_id == 248')['device_id_hash'].nunique()
```

```
In [29]: # разница в числе пользователей контрольных групп
users_diff = difference(unique_users_246, unique_users_247)
print(f'Разница в числе пользователей между контрольными группами:\n{users_diff[0]} пользователей или {users_diff[1]}%.')
```

Разница в числе пользователей между контрольными группами: 29 пользователей или 1.2%.

Как выяснили ранее, пользователи уникальны для каждой группы.

Самое популярное событие

Согласно воронке самое популярное событие MainScreenAppear для всех групп. Изучим число пользователей, открывших основной экран в каждой группе.

Для сравнения долей пользователей (конверсии) в разных группах будем применять **z-test**.

Нулевая и альтернативная гипотезы:

$$\begin{cases} H_0 : \text{Между долями нет значимой разницы.} \\ H_1 : \text{Между долями есть значимая разница.} \end{cases}$$

```
In [30]: for group in events_funnel_groups['exp_id'].sort_values().unique():
        popular_event_users = (
            events_funnel_groups
            .query('exp_id == @group & event_name == "MainScreenAppear"')
            ['unique_users'].values[0]
        )
        unique_users_group = data_upd.query('exp_id == @group')['device_id_hash'].nunique()
        print(f'Группа {group}')
        print(f'Число пользователей, открывших основной экран: {popular_event_users}.')
        print(f'Доля пользователей, открывших основной экран: {popular_event_users / unique_')
        print()
        print('=====')
```

Группа 246

Число пользователей, открывших основной экран: 2450.

Доля пользователей, открывших основной экран: 0.986

=====

Группа 247

Число пользователей, открывших основной экран: 2476.

Доля пользователей, открывших основной экран: 0.985

=====

Группа 248

Число пользователей, открывших основной экран: 2493.

Доля пользователей, открывших основной экран: 0.983

=====

Проверим гипотезу о равенстве долей (по уникальным пользователям) в 246 и 247 группах.

Рассмотрим доли пользователей, совершивших покупку, от пользователей, посетивших основной экран.

```
In [31]: # проверка гипотезы
payment_screen_246 = (
    events_funnel_groups
    .query('exp_id == 246 & event_name == "PaymentScreenSuccessful"')
    ['unique_users'].values[0])
payment_screen_247 = (
    events_funnel_groups
    .query('exp_id == 247 & event_name == "PaymentScreenSuccessful"')
    ['unique_users'].values[0])
main_screen_246 = (
    events_funnel_groups
    .query('exp_id == 246 & event_name == "MainScreenAppear"')
    ['unique_users'].values[0])
main_screen_247 = (
    events_funnel_groups
    .query('exp_id == 247 & event_name == "MainScreenAppear"')
    ['unique_users'].values[0])

check_group_diff(payment_screen_246, payment_screen_247, main_screen_246, main_screen_247)
```

p-значение: 0.12044299485641763

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Вывод

При сравнении долей пользователей между группам, дошедших с основного экрана до экрана

завершённой покупки, нет основания считать их разными.

Проверка контрольных групп

Проведём проверку равенства долей пользователей для каждого события.

```
In [32]: # проверка гипотезы о равенстве долей пользователей для каждого события
for event in events_funnel_groups['event_name'].unique():
    event_users_246 = (
        events_funnel_groups
        .query('exp_id == 246 & event_name == @event')
        ['unique_users'].values[0])
    event_users_247 = (
        events_funnel_groups
        .query('exp_id == 247 & event_name == @event')
        ['unique_users'].values[0])
    print()
    print(f'Событие "{event}":')
    print()
    print(f'Число пользователей в группе 246: {event_users_246}.')
    print(f'Число пользователей в группе 247: {event_users_247}.')
    check_group_diff(event_users_246, event_users_247, unique_users_246, unique_users_247)
    print('=' * 80)
```

Событие "MainScreenAppear":

Число пользователей в группе 246: 2450.

Число пользователей в группе 247: 2476.

р-значение: 0.7570597232046099

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие "OffersScreenAppear":

Число пользователей в группе 246: 1542.

Число пользователей в группе 247: 1520.

р-значение: 0.2480954578522181

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие "CartScreenAppear":

Число пользователей в группе 246: 1266.

Число пользователей в группе 247: 1238.

р-значение: 0.22883372237997213

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие "PaymentScreenSuccessful":

Число пользователей в группе 246: 1200.

Число пользователей в группе 247: 1158.

р-значение: 0.11456679313141849

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Вывод

По каждому событию контрольные группы не имеют статистических различий.

Можем считать разбиение на группы **корректным**.

Группа с изменённым шрифтом

```
In [33]: print(f'Число пользователей в тестируемой группе: {unique_users_248}.')
```

Число пользователей в тестируемой группе: 2537.

Сравнение тестовой группы с контрольными

В следующей ячейке проверка разницы между тестовой группой и контрольными группами.

Адаптирован алгоритм из предыдущего раздела.

```
In [34]: # сравнение тестовой группы с контрольными
for group in [246, 247]:
    print()
    print(f'Сравнение тестовой группы 248 с контрольной группой {group}.')
    for event in events_funnel_groups['event_name'].unique():
        # отбираем тестовую группу 248
        event_users_248 = (
            events_funnel_groups
            .query('exp_id == 248 & event_name == @event')
            ['unique_users'].values[0])

        # отбираем контрольную группу @group
        event_users_gr = (
            events_funnel_groups
            .query('exp_id == @group & event_name == @event')
            ['unique_users'].values[0])

    print()
    print(f'Событие "{event}":')
    print()
    print(f'Число пользователей в группе 248: {event_users_248}.')
    print(f'Число пользователей в группе {group}: {event_users_gr}.')
    print()

    # выбор размера группы для check_group_diff
    if group == 246:
        unique_users_gr = unique_users_246
    if group == 247:
        unique_users_gr = unique_users_247

    # проверка гипотезы о равенстве долей
    check_group_diff(event_users_gr, event_users_248, unique_users_gr, unique_users_248)
    print('=' * 80)

print()
print('=' * 80)
print('=' * 80)
```

Сравнение тестовой группы 248 с контрольной группой 246.

Событие "MainScreenAppear":

Число пользователей в группе 248: 2493.

Число пользователей в группе 246: 2450.

p-значение: 0.2949721933554552

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие "OffersScreenAppear":

Число пользователей в группе 248: 1531.

Число пользователей в группе 246: 1542.

p-значение: 0.20836205402738917

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие "CartScreenAppear":

Число пользователей в группе 248: 1230.
Число пользователей в группе 246: 1266.

p-значение: 0.07842923237520116

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие "PaymentScreenSuccessful":

Число пользователей в группе 248: 1181.
Число пользователей в группе 246: 1200.

p-значение: 0.2122553275697796

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Сравнение тестовой группы 248 с контрольной группой 247.

Событие "MainScreenAppear":

Число пользователей в группе 248: 2493.
Число пользователей в группе 247: 2476.

p-значение: 0.4587053616621515

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие "OffersScreenAppear":

Число пользователей в группе 248: 1531.
Число пользователей в группе 247: 1520.

p-значение: 0.9197817830592261

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие "CartScreenAppear":

Число пользователей в группе 248: 1230.
Число пользователей в группе 247: 1238.

p-значение: 0.5786197879539783

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Событие "PaymentScreenSuccessful":

Число пользователей в группе 248: 1181.
Число пользователей в группе 247: 1158.

p-значение: 0.7373415053803964

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Вывод

При попарном сравнении долей между тестируемой и контрольными группами для каждого события не нашлось статистически значимых различий в этих группах.

Сравнение тестовой группы с объединёнными контрольными

```
In [35]: # сравнение тестовой группы с объединёнными контрольными
for event in events_funnel_groups['event_name'].unique():
    event_users_control = (
        events_funnel_groups
        .query('exp_id != 248 & event_name == @event')
        ['unique_users'].sum())
    event_users_248 = (
        events_funnel_groups
        .query('exp_id == 248 & event_name == @event')
        ['unique_users'].values[0])
    print()
    print(f'Событие "{event}":')
    print()
    print(f'Число пользователей в объединённой контрольной группе: {event_users_control}')
    print(f'Число пользователей в тестовой группе 248: {event_users_248}.')
    check_group_diff(event_users_control, event_users_248, unique_users_246 + unique_users_247)
    print('=' * 80)
```

Событие "MainScreenAppear":

Число пользователей в объединённой контрольной группе: 4926.
Число пользователей в тестовой группе 248: 2493.
p-значение: 0.29424526837179577
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
=====

Событие "OffersScreenAppear":

Число пользователей в объединённой контрольной группе: 3062.
Число пользователей в тестовой группе 248: 1531.
p-значение: 0.43425549655188256
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
=====

Событие "CartScreenAppear":

Число пользователей в объединённой контрольной группе: 2504.
Число пользователей в тестовой группе 248: 1230.
p-значение: 0.18175875284404386
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
=====

Событие "PaymentScreenSuccessful":

Число пользователей в объединённой контрольной группе: 2358.
Число пользователей в тестовой группе 248: 1181.
p-значение: 0.6004294282308704
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
=====

Выводы:

- нет оснований считать доли в группах разными;
- деление по группам для A/B теста между контрольными группами 246 и 247, тестовой группой 248 проведено корректно.

Поправка на множественную проверку гипотез

При проверке статистических гипотез был выбран уровень значимости $\alpha = 0.05$.

Было проведено 17 проверок статистических гипотез.

```
In [36]: print(f'Вероятность получения ложнопозитивного результата: {prob_check(.05, 17) * 100:.1
```

Вероятность получения ложнопозитивного результата: 58.2%

Нет смысла проводить проверки ещё раз, т.к. все ранее полученные p-value больше 0.05.

При уменьшении уровня значимости p-value всё ещё будут больше.

Можно уменьшить уровень значимости до 0.005, чтобы при 17 тестах вероятность ложнопозитивного результата была около 8%.

```
In [37]: print(f'Вероятность получения ложнопозитивного результата: {prob_check(.005, 17) * 100:.1
```

Вероятность получения ложнопозитивного результата: 8.2%

Заключение и выводы

Заключение

В работе был исследован журнал событий с 1 по 7 августа.

Составлена воронка продаж, по которой было проверено разбиение пользователей на группы.

Проверены результаты A/A/B-теста за указанный период.

Выводы

- Разделение по группам проведено **корректно**, группы не имеют статистических различий.
- Для проверки долей **достаточен** уровень статистической значимости в 5%.
- Принимая во внимание **отсутствие разницы между группами** в каждом из тестов, можно констатировать, что установка нового шрифта не влияет на конверсию в покупки.

Рекомендации

Поведение пользователей тестовой группы не отличается от поведения пользователей контрольных групп.

Нет поводов отказаться от внедрения нового шрифта.