Содержание работы

В данн работе нам необходимо изучить данные пользователей мобильного приложения продаж. Исследование состоит из двух этапов:

Этап 1: Исследование воронки продаж. Необходимо выяснить, как пользователи проходят по этапам покупки, и на каких стадия происходит потеря пользователя.

Этап 2: Исследование результатов A/A/B теста. Необходимо выяснить, приведет ли изменение в интерфейсе к увеличению продаж.

Описание данных: Каждая запись в таблице — это действие пользователя, или событие.

EventName — название события; DeviceIDHash — уникальный идентификатор пользователя; EventTimestamp — время события; ExpId — номер эксперимента: 246 и 247 — контрольные группы, а 248 — экспериментальная.

Импорт библиотек и данных

Ввод [1]:
```
import pandas as pd
import numpy as np
import datetime as dt
import plotly.express as px
from plotly import graph_objects as go
from plotly.subplots import make_subplots
from statsmodels.stats.proportion import proportions_ztest, proportion_confint
```

Ввод [3]:
```
df = pd.read_csv('D:\download\проект\logs_exp.csv','\t')
```

Проверка данных

Ввод [4]:
```
df.info()
```

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

Пустых значений нет, с типами данных согласен.

Ввод [5]:
```
display(df.sample(10))
```

	EventName	DeviceIDHash	EventTimestamp	ExpId
46723	MainScreenAppear	6274455082583969109	1564732395	246
47949	MainScreenAppear	1225191038684666184	1564734628	247
10587	OffersScreenAppear	1631833249019071459	1564648603	246
23592	MainScreenAppear	3722204315779155652	1564669136	247
82982	OffersScreenAppear	4779085621780049711	1564823602	248
91799	OffersScreenAppear	6510083166221201809	1564836840	248
219032	MainScreenAppear	1730229860931366431	1565163814	246
181286	CartScreenAppear	1754140665440434215	1565070595	247
156439	MainScreenAppear	670503647908011387	1565007167	247
49548	MainScreenAppear	6326495816973010604	1564737531	247

В принципе EventName и ExpId можно поменять тип данных на категориальный, но это не принципиально. Переименую столбцы исходя из моих представлений об удобстве именования

Ввод [6]:
```
#df.columns=['ev_name','ev_userid','ev_timest','ev_group']
df = df.rename(columns={'EventName': 'ev_name', 'DeviceIDHash': 'ev_userid','EventTimestamp':'ev_timest','ExpId':'ev_group'})
```

Посмотрим сколько у на пользователей в разных группах, и какие вообще группы есть. И определя переменную groups, которую в дальнейшем буду использовать для итерации по группам.

Ввод [7]:
```
print(df.groupby('ev_group').agg({'ev_userid':'count'}))
groups = [246, 247, 248]
```

```
          ev_userid
ev_group
246           80304
247           78075
248           85747
```

Групп для A/A/B теста три, как и должно быть. Число пользователей контрольных A/A групп различается на 3 процента, а разница между группой A номер 247 и группой B около 9%. Многовато, но еще допустимо.

Выделю время из столбца EventTimestamp

Ввод [8]:
```
df['ev_dt']=df.ev_timest
df.ev_dt=pd.to_datetime(df['ev_dt'], unit='s')
df['ev_date']=df.ev_dt.dt.date
df.ev_date=pd.to_datetime(df.ev_date)
```

Посмотрим, какие типы событий етсь в данных

Ввод [9]:
```
print(df.ev_name.unique())
```

```
['MainScreenAppear' 'PaymentScreenSuccessful' 'CartScreenAppear'
 'OffersScreenAppear' 'Tutorial']
```

Количество событий по типам:

Ввод [10]:
```
print(df.groupby('ev_name').agg({'ev_userid':'count'}))
```

```
                         ev_userid
ev_name
CartScreenAppear             42731
MainScreenAppear            119205
OffersScreenAppear           46825
PaymentScreenSuccessful      34313
Tutorial                      1052
```

Сколько всего уникальных пользователей

Ввод [11]:
```
print("уникальных пользователей : {}".format(len(df.ev_userid.unique())))
```

```
уникальных пользователей : 7551
```

Проверим данные на дубликаты. Проверку и удаление выполню, взяв для анализ все столбцы кроме номера группы. Таким образом, я уберу случаи, когда один и тот-же пользователь попал в несколько групп A/B теста.

Ввод [12]:
```
print(df.duplicated(subset=['ev_name', 'ev_userid', 'ev_timest']).sum())
```

```
413
```

Имеется 413 дубликатов, удаляем.

Ввод [13]:
```
df=df.drop_duplicates(subset=['ev_name', 'ev_userid', 'ev_timest'])
```

Посмотрим, что у нас с датами

Ввод [14]:
```
print(df.ev_dt.describe())
```

```
count                  243713
unique                 176654
top       2019-08-01 14:40:35
freq                        9
first     2019-07-25 04:43:36
last      2019-08-07 21:15:17
Name: ev_dt, dtype: object
```

```
<ipython-input-14-19cd4559111f>:1: FutureWarning: Treating datetime data as categorical rather than numeric in `.describe` is deprecated and will be removed in a future version of pandas. Specify `datetime_is_numeric=True` to silence this warning and adopt the future behavior now.
  print(df.ev_dt.describe())
```

Данные за период с 25.07.19 по 07.08.19. Посмотри, что с распределение числа событий по дням.

Ввод [15]:
```
dftmp=df.groupby('ev_date').agg({'ev_name':'count'}).reset_index()

fig = go.Figure(data=[go.Bar(x=dftmp['ev_date'], y=dftmp['ev_name'],text=dftmp['ev_name'],textposition='auto')])
fig.update_xaxes(title_text='Дата')
fig.update_yaxes(title_text='Количество событий')
fig.update_layout(title_text='Количество событий за весь период')
fig.show()
```

Данные за период до 01.07 отброшу, т.к. их можно считать недостоверными, и они составляю менее 1% от всех данных.

Ввод [16]:
```
print('Число записей с датой <=31.07.2019 : {}'.format(len(df[df.ev_date<='2019-07-31'].index)))
print('Количество уникальных пользователей в периоде до 31.07.2019 : {}'.format(df[df.ev_date<='2019-07-31'].ev_userid.nunique()))
```

```
Число записей с датой <=31.07.2019 : 2826
Количество уникальных пользователей в периоде до 31.07.2019 : 1451
```

Ввод [17]:
```
# формируем список пользователей в удаляемых данных
dfdel=df[df.ev_date<='2019-07-31'].groupby('ev_userid').agg({'ev_name':'count'}).reset_index()
```

Ввод [18]:
```
df.drop(df[df.ev_date<='2019-07-31'].index,inplace=True)
df=df.reset_index(drop=True)
```

Ввод [19]:
```
x=dfdel.ev_userid.isin(df.ev_userid)
display(x.value_counts())
```

```
True     1434
False      17
Name: ev_userid, dtype: int64
```

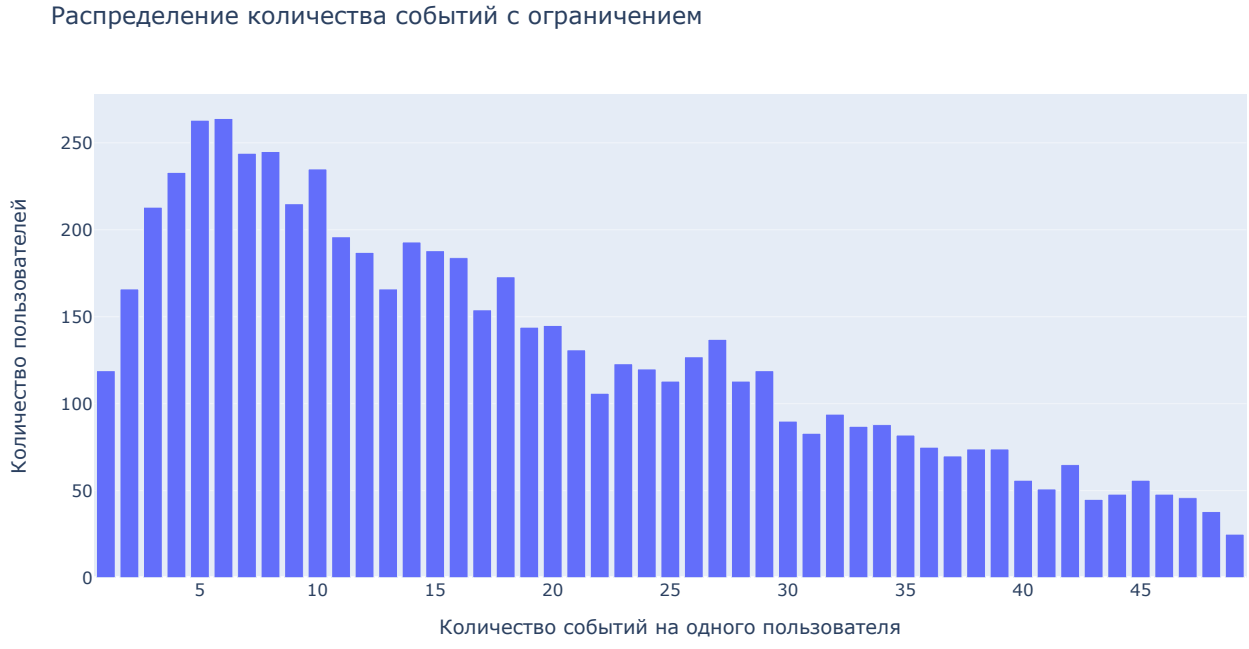
Посмотрим, сколько событий приходиться на одного пользователя.

Ввод [20]:
```
dftmp=df.groupby('ev_userid').agg({'ev_name':'count'}).reset_index()
print(dftmp.ev_name.describe())
ddf=dftmp.groupby('ev_name').agg({'ev_userid':'count'}).reset_index()

fig = go.Figure(data=[go.Bar(x=ddf['ev_name'], y=ddf['ev_userid'])])
fig.update_xaxes(title_text='Количество событий на одного пользователя')
fig.update_yaxes(title_text='Количество пользователей')
fig.update_layout(title_text='Распределение количества событий')
fig.show()
```

```
count    7534.000000
mean       31.973321
std        65.090307
min         1.000000
25%         9.000000
50%        19.000000
75%        37.000000
max      2307.000000
Name: ev_name, dtype: float64
```

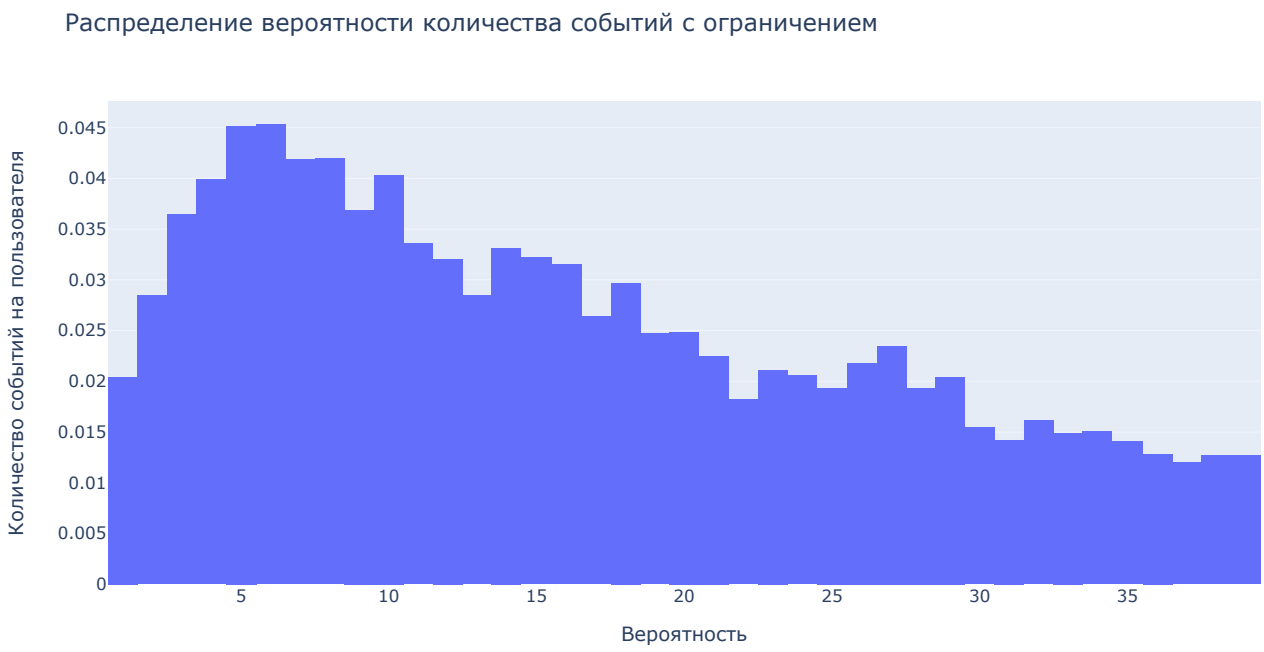
Слишком длинный хвост, построю то-же самое но для пользователей с менее чем 50 событий на пользователя.

Ввод [21]:
```
ddf=ddf[ddf.ev_name<50]
fig = go.Figure(data=[go.Bar(x=ddf['ev_name'], y=ddf['ev_userid'])])
fig.update_xaxes(title_text='Количество событий на одного пользователя')
fig.update_yaxes(title_text='Количество пользователей')
fig.update_layout(title_text='Распределение количества событий с ограничением ')
fig.show()
```



75% процентов пользователей имеют менее 40 событий. Посмотрим на плотность распределения. Для плотности я сразу возьму пользователей с числом событий менее 40, иначе это будет не колокол а дельта-функция и график будет не информативный.

Ввод [22]:
```
fig = go.Figure(data=[go.Histogram(x=dftmp[dftmp.ev_name<40]['ev_name'],histnorm='probability density')])
fig.update_xaxes(title_text='Вероятность')
fig.update_yaxes(title_text='Количество событий на пользователя')
fig.update_layout(title_text='Распределение вероятности количества событий с ограничением ')
fig.show()
```

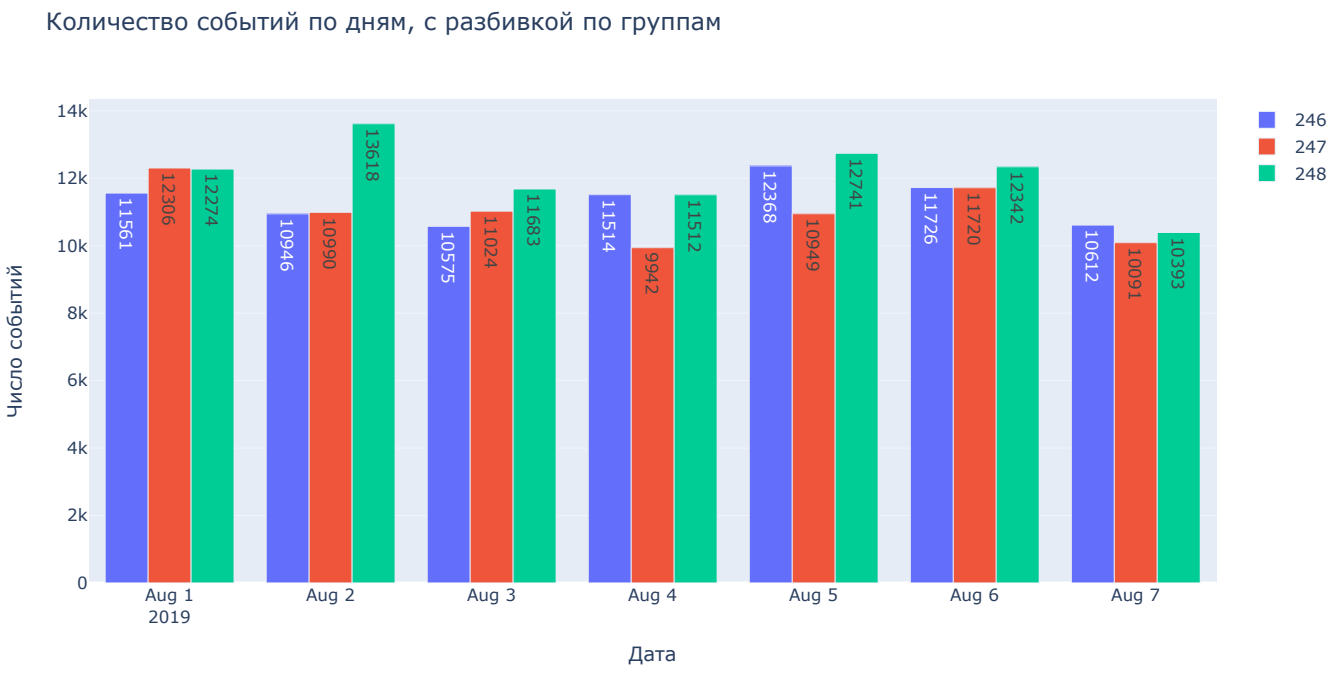


Среднее было 31, а судя по графику плотности, наиболее часто встречающееся значение числа событий на пользователя - около 6.

Посмотрю еще раз на разбивку количества событий по дням, после удаления данных до 01.08.19

Ввод [23]:
```
dftmp=df.groupby('ev_date').agg({'ev_name':'count'}).reset_index()

fig = go.Figure(data=[go.Bar(x=dftmp['ev_date'], y=dftmp['ev_name'],text=dftmp['ev_name'],textposition='auto')])
fig.update_xaxes(title_text='Дата')
fig.update_yaxes(title_text='Количество событий')
fig.update_layout(title_text='Количесиво событий по дням, после удаления')
fig.show()
```

Ввод [24]:
```
fig = go.Figure()
for el in groups:
    dftmp=df[df.ev_group==el].groupby(['ev_date']).ev_userid.agg(['count']).sort_values(by='count',ascending=False)
    fig.add_trace(go.Bar(
        name=el,
        x=dftmp.index,
        y=dftmp['count'],
        text=dftmp['count'],
        textposition='auto'))

fig.update_xaxes(title_text='Дата')
fig.update_yaxes(title_text='Число событий')
fig.update_layout(title_text='Количество событий по дням, с разбивкой по группам')

fig.show()
```



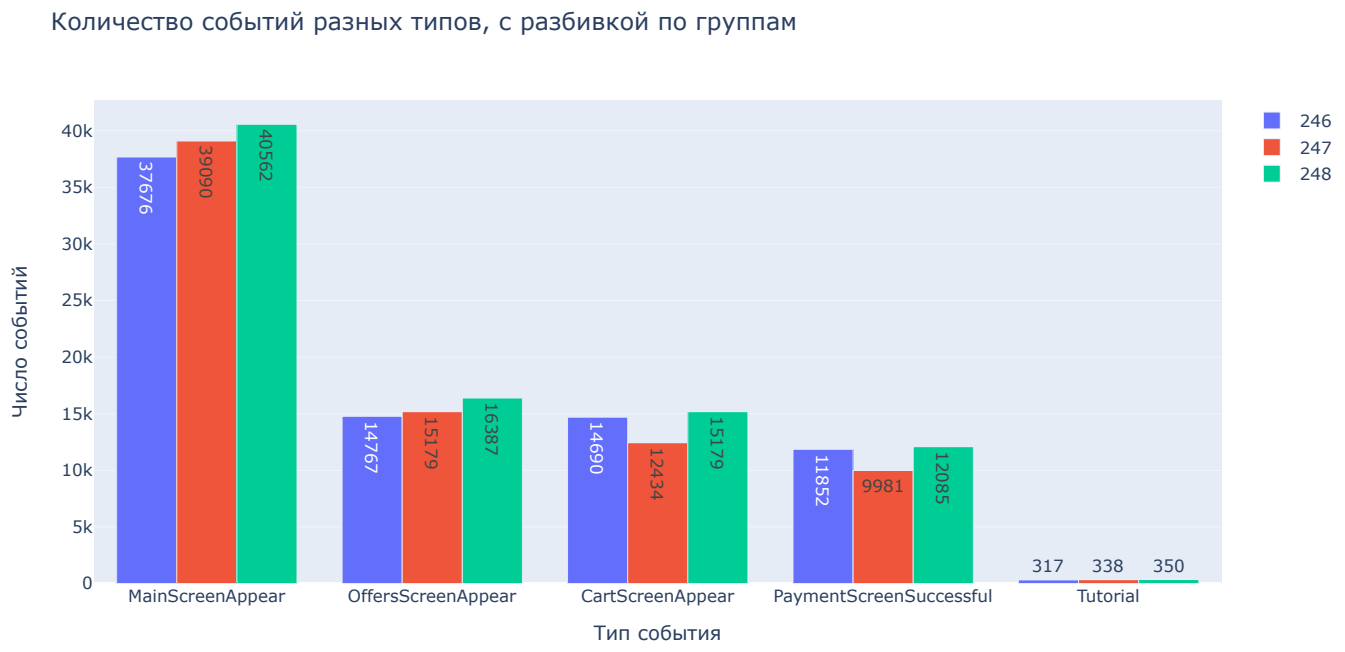
Разброс есть, но аномалий не вижу.

Анализ воронки событий

Посмотрим, как часто встречаются события разных типов, с разбивкой по группам.

типы и частота событий

Ввод [25]:
```
#groups - переменная, содержащая номера групп

fig=go.Figure()
for el in groups:
    dftmp=df[df.ev_group==el].groupby(['ev_name']).ev_userid.agg(['count']).sort_values(by='count',ascending=False)
    fig.add_trace(go.Bar(
        name=el,
        x=dftmp.index,
        y=dftmp['count'],
        text=dftmp['count'],
        textposition='auto'))

fig.update_xaxes(title_text='Тип события')
fig.update_yaxes(title_text='Число событий')
fig.update_layout(title_text='Количество событий разных типов, с разбивкой по группам')
fig.show()
```



То-же самое, но без разбивки по группам

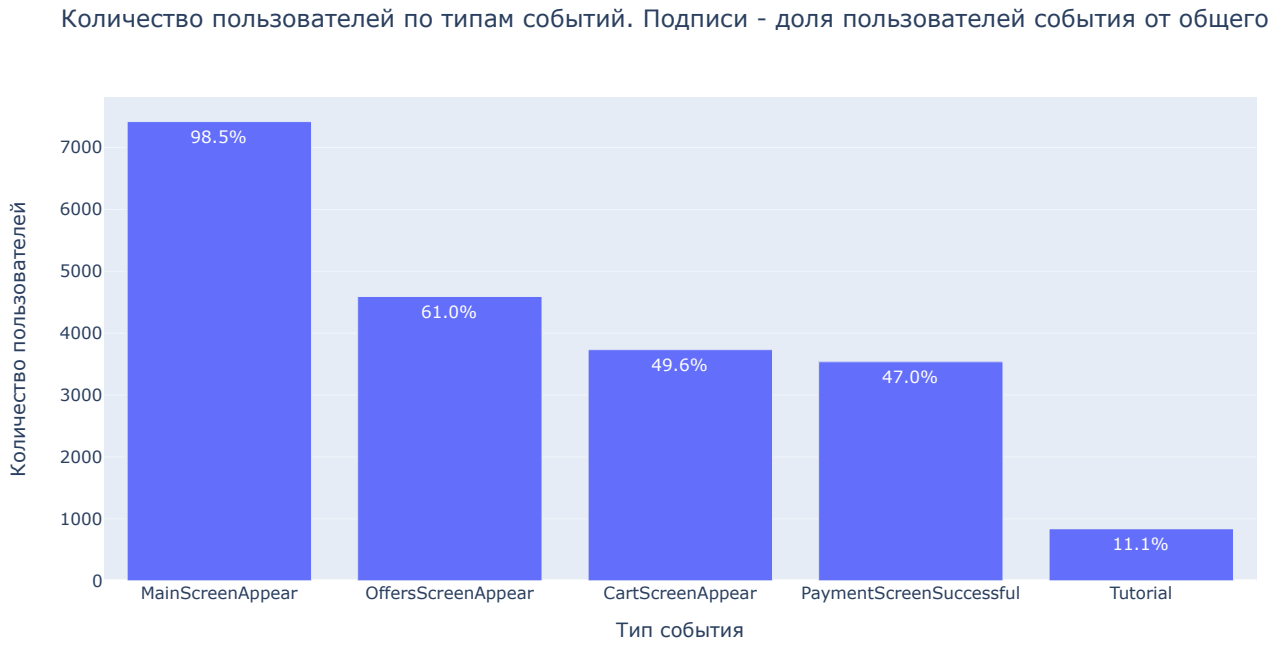
Ввод [26]:
```
dftmp=df.groupby('ev_name').ev_userid.agg(['count','nunique']).sort_values(by='count',ascending=False)
fig = go.Figure(data=[go.Bar(x=dftmp.index, y=dftmp['count'],text=dftmp['count'],textposition='auto')])
fig.update_xaxes(title_text='Тип события')
fig.update_yaxes(title_text='Число событий')
fig.update_layout(title_text='Количество событий разных типов')
fig.show()
```

Вот и воронка событий, а обучение потребовалось менее 0,01% пользователей.

пользователей на событие

Посмотрим сколько уникальны пользователей совершали то или иное событие.

Ввод [27]:
```
#вычисляем процент пользователей, хоть раз совершивших действие.
total_users=len(df.ev_userid.unique())
bar_text=dftmp['nunique']/total_users
bar_text=bar_text.apply(lambda s: '{:.1%}'.format(s))


#рисуем график
fig = go.Figure(data=[go.Bar(x=dftmp.index, y=dftmp['nunique'],text=bar_text,textposition='inside')])

fig.update_xaxes(title_text='Тип события')
fig.update_yaxes(title_text='Количество пользователей')
fig.update_layout(title_text='Количество пользователей по типам событий. Подписи - доля пользователей события от общего числа.')
fig.show()
```



Посмотрим на то-же самое с разбивкой по группам.

Ввод [28]:
```
fig=go.Figure()
for el in groups:
    dftmp=df[df.ev_group==el].groupby(['ev_name']).ev_userid.agg(['nunique']).sort_values(by='nunique',ascending=False)
    total_users=len(df[df.ev_group==el].ev_userid.unique())
    bar_text=dftmp['nunique']/total_users
    bar_text=bar_text.apply(lambda s: '{:.1%}'.format(s))
    fig.add_trace(go.Bar(
        name=el,
        x=dftmp.index,
        y=dftmp['nunique'],
        text=bar_text,
        textposition='auto'))

fig.update_xaxes(title_text='Тип события')
fig.update_yaxes(title_text='Количество пользователей')
fig.update_layout(title_text='Количество пользователей по типам событий. Подписи - доля пользователей события от общего числа.')
fig.show()
```

Аномалий нет, по событиям - воронка, по группам - примерно одинаково.

порядок событий

Посмотрев на типы событий, можно принять следующее решение: Воронка событий будет представленна в следующем виде: MainScreenAppear -> OffersScreenAppear -> CartScreenAppear -> PaymentScreenSuccessful

Событие Tutorial нужно исключить из воронки событий, т.к. оно не имеет непосредственного участия в процессе покупки.

воронка событий

Приступим к визуализации воронки событий. Подготовим данные.

Ввод [29]:
```
df_funel=df[df.ev_name!='Tutorial'].loc[:,['ev_name','ev_userid','ev_group']]
dftmp=df_funel.groupby(['ev_name','ev_group']).agg({'ev_userid':'nunique'}).reset_index()
dftmp=dftmp.sort_values(by='ev_userid',ascending=False)
```

Построим воронку событий с разбивкой по группам, но без учета последовательности переходов. Т.е. берем всех уникальных пользователей каждого события.

Ввод [30]:
```
fig=go.Figure()
y_labels=pd.Series(dftmp.ev_name.unique())
for i, group in enumerate(groups):
    fig.add_trace(go.Funnel(
        name=str(group),
        x=dftmp[dftmp.ev_group==group].ev_userid,
        y=y_labels,  #y_labels - моя переменная, в которой в нужном порядке перечисленны типы событий
        textinfo = "value+percent previous",
        connector = {"fillcolor": '#bde0eb'},
        insidetextfont = {'color': 'white', 'size': 14}))

fig.update_yaxes(title_text='Уникальных пользователей')
fig.update_layout(
    title_text="Воронка событий без учета последовательности переходов",
    height=600,
    legend=dict(
    orientation="h",
    bgcolor = 'rgba(0,0,0,0.05)',
    yanchor="top",
    y=0,
    xanchor="left",
    x=0.01))


fig.show()
```

В целом выглядит не плохо. 63% решили сделать покупку, 82% выбрали товар и 95% оплатили. Теперь посмотрим воронку событий с учетом последовательности переходов для каждого пользователя.

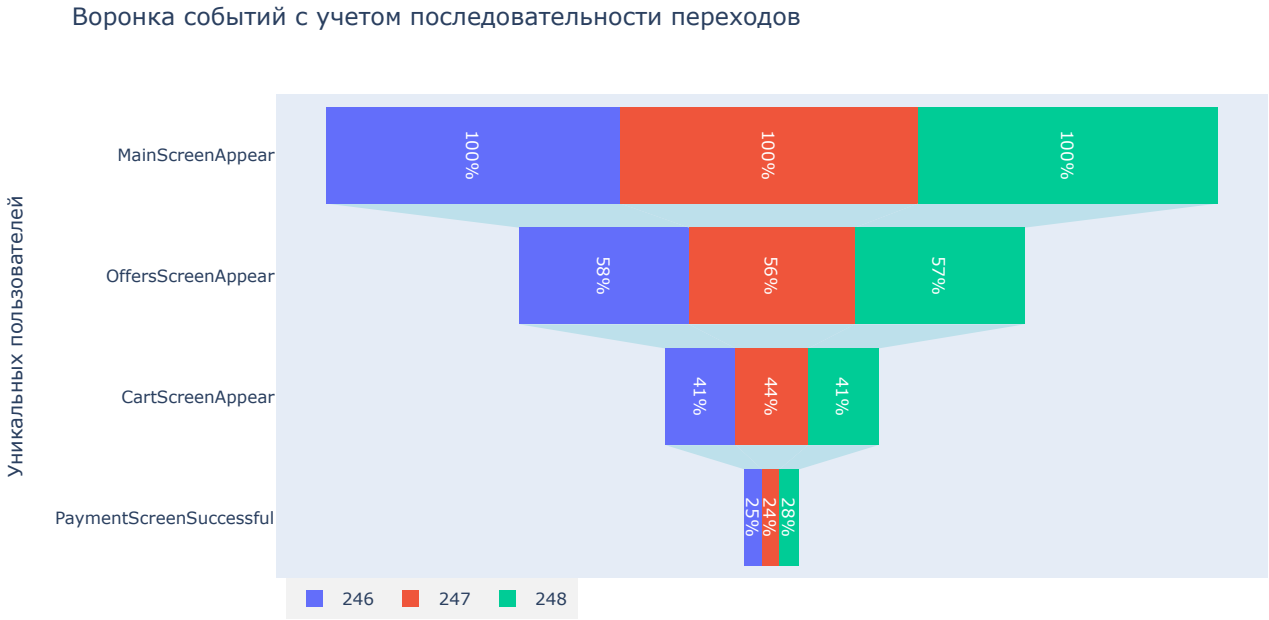
Подготовка данных.

Ввод [31]:
```
users = df[df['ev_name'] != 'Tutorial'].pivot_table(
        index=['ev_userid', 'ev_group'],
        columns='ev_name',
        values='ev_dt',
        aggfunc='min').reset_index()

funnel ={}
for group in groups: #groups - переменная в которой перечисленны все группы
    funnel[group] =[]
    step_1 = (users['ev_group']==group) & (~users['MainScreenAppear'].isna())
    step_2 = step_1 & (users['OffersScreenAppear'] > users['MainScreenAppear'])
    step_3 = step_2 & (users['CartScreenAppear'] > users['OffersScreenAppear'])
    step_4 = step_3 & (users['PaymentScreenSuccessful'] > users['CartScreenAppear'])
    funnel[group].append(users[step_1].shape[0])
    funnel[group].append(users[step_2].shape[0])
    funnel[group].append(users[step_3].shape[0])
    funnel[group].append(users[step_4].shape[0])
```

Визуализация

Ввод [32]:

```
fig = go.Figure()
for i, group in enumerate(groups):
    fig.add_trace(go.Funnel(
        name = str(group),
        y = y_labels, #y_labels - моя переменная, в которой в нужном порядке перечисленны типы событий
        x = funnel[group],
        textposition = "inside",
        textinfo = "percent previous",
        constraintext='outside',
        textangle = 90,
        connector = {"fillcolor": '#bde0eb'},
        insidetextfont = {'color': 'white'}))
fig.update_yaxes(title_text='Уникальных пользователей')
fig.update_layout(
    title_text="Воронка событий с учетом последовательности переходов",
    legend=dict(
    orientation="h",
    bgcolor = 'rgba(0,0,0,0.05)',
    yanchor="top",
    y=0,
    xanchor="left",
    x=0.01))


fig.show()
```



Предполагаемой последовательности шагов следует многие пользователи, но далеко не все. Это нам говорит о том, что есть несколько способов дойти до оплаты в приложении, например возможность мгновенной оплаты товара без перехода в корзину.

Анализ теста

dftmp=df.groupby('ev_group').agg({'ev_userid':'nunique'}).reset_index() print(dftmp)

dft=pd.crosstab(df.ev_userid,df.ev_group) dft.columns=['A1','A2','B'] dftAll=dft[(dft.A1!=0) & (dft.B!=0) & (dft.A2!=0)] dftAll.info()

dftAll=dft[((dft.A1!=0) | (dft.A2!=0)) & (dft.B!=0)] dftAll.info()

Приступим к анализу результатов A/A/B теста

проверка A/A групп

Сначала проверим данные обоих групп A, что-бы удостовериться, что они соответствую друг другу. Проверять гипотезы на статистическую значимость мы будем с помощью z-теста.

Подготовим данные.

Ввод [33]:
```
# переменная ztest_df будет содержать данные для анализа
ztest_df=df[df.ev_name!='Tutorial'].pivot_table(
 index='ev_name',
 columns='ev_group',
 values='ev_userid',
 aggfunc='nunique')

# добавим столбец, содержащий сумму данных обеих групп A
ztest_df['All']=ztest_df[246]+ztest_df[247]

# вычисляем общее число событий
tmpdf=df[df.ev_name!='Tutorial'].groupby('ev_group').agg({'ev_userid':'nunique'}).transpose()

# добавим столбец, содержащий сумму данных обеих групп A
tmpdf['All']=tmpdf[246]+tmpdf[247]

# добавляем строку содержащую общее число событий
ztest_df=ztest_df.append(tmpdf)

# переименуем название итоговых цифр
ztest_df=ztest_df.rename(index={'ev_userid':'total'})
```

Ввод [34]:
```
display(ztest_df)
```

ev_group	246	247	248	All
CartScreenAppear	1266	1238	1230	2504
MainScreenAppear	2450	2476	2493	4926
OffersScreenAppear	1542	1520	1531	3062
PaymentScreenSuccessful	1200	1158	1181	2358
total	2483	2512	2535	4995

Как представленно выше, переменная ztest_df содержит данные положительных исходов и общее число событий в разрезе типов событий и групп. Столбец All - это сумма столбцов 246 и 247. Строка total - это общее число событий.

Оформим блок, вычисляющий статистическую разницу и отображающий графики в виде процедуры.

Ввод [35]:

```
def z_test(group_a,group_b,alpha): # параметры вызова: группа 1, группа 2, пороговое значение стат. значимости

    count_a=ztest_df[group_a].total # выбираем значения всех исходов, переданных в параметрах вызова групп
    count_b=ztest_df[group_b].total

    for el in y_labels: #y_labels - моя переменная, в которой в нужном порядке перечисленны типы событий
        success_a=ztest_df[ztest_df.index==el][group_a].to_numpy()   # выбираем значения положительных исходов,
        success_b=ztest_df[ztest_df.index==el][group_b].to_numpy()   # для групп переданных в параметрах вызова

        successes = np.hstack([success_a, success_b])                # формируем структуры, для передачи в функцию,
        nobs = [count_a, count_b]                                    # вычисляющую z-тест

        z_stat, pval = proportions_ztest(successes, nobs=nobs)      # вычисляем z-тест

        # я заодно вычислил еще и эти дополнительные параметры, что-бы посмотреть как это работает,
        # но незнаю, стоит ли их оставлять в итоговом решении.
        #(lower_con, lower_treat), (upper_con, upper_treat) = proportion_confint(successes, nobs=nobs, alpha=alpha)
        print('Группа {} p-value: {:.3f}'.format(el,pval))
        if (pval < alpha):
            print("Отвергаем нулевую гипотезу: между долями есть значимая разница")
        else:
            print("Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными")
        print('')

    # рисуем воронки. Можно было сделать в цикле с перебором кортежа из входных параметров процедуры, но я оставил так.
    fig = go.Figure()
    funel_df=ztest_df[ztest_df.index!='total'].sort_values(by=246,ascending=False)
                            # воронка для группы 1
    fig.add_trace(go.Funnel(
            name=str(group),
            x=funel_df[group_a],
            y=y_labels,
            textinfo = "value+percent previous",
            connector = {"fillcolor": '#bde0eb'},
            insidetextfont = {'color': 'white', 'size': 14}))
                            # воронка дл ягруппы 2
    fig.add_trace(go.Funnel(
            name=str(group),
            x=funel_df[group_b],
            y=y_labels,
            textinfo = "value+percent previous",
            connector = {"fillcolor": '#bde0eb'},
            insidetextfont = {'color': 'white', 'size': 14}))

    fig.update_yaxes(title_text='Уникальных пользователей')
    fig.update_layout(
    title_text="Воронка событий без учета последовательности переходов",
    height=600,
    legend=dict(
    orientation="h",
    bgcolor = 'rgba(0,0,0,0.05)',
    yanchor="top",
    y=0,
    xanchor="left",
    x=0.01))
    fig.show()
```

Проверяем статистические критерии разницы между контрольными группами A1 и A2.

Считая распределение генеральной совокупности нормальным:

1. Нулевая гипотеза: доли удачных исходов двух групп равны.
2. Альтернативная гипотеза: доли удачных исходов двух групп не равны.

Ввод [36]:
```
z_test(246,247,0.05)
```

```
Группа MainScreenAppear p-value: 0.753
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Группа OffersScreenAppear p-value: 0.248
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Группа CartScreenAppear p-value: 0.229
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Группа PaymentScreenSuccessful p-value: 0.114
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

проверка группы с изменениями

Проверяем статистические критерии разницы между контрольной группой A1 и группой B.

Считая распределение генеральной совокупности нормальным:

1. Нулевая гипотеза: доли удачных исходов двух групп равны.
2. Альтернативная гипотеза: доли удачных исходов двух групп не равны.

Ввод [37]:
```
z_test(246,248,0.05)
```

```
Группа MainScreenAppear p-value: 0.339
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Группа OffersScreenAppear p-value: 0.214
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Группа CartScreenAppear p-value: 0.081
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Группа PaymentScreenSuccessful p-value: 0.217
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

Проверяем статистические критерии разницы между контрольной группой A2 и группой B.

Считая распределение генеральной совокупности нормальным:

1. Нулевая гипотеза: доли удачных исходов двух групп равны.
2. Альтернативная гипотеза: доли удачных исходов двух групп не равны.

Ввод [38]:
```
z_test(247,248,0.05)
```

```
Группа MainScreenAppear p-value: 0.519
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Группа OffersScreenAppear p-value: 0.933
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Группа CartScreenAppear p-value: 0.588
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Группа PaymentScreenSuccessful p-value: 0.728
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```

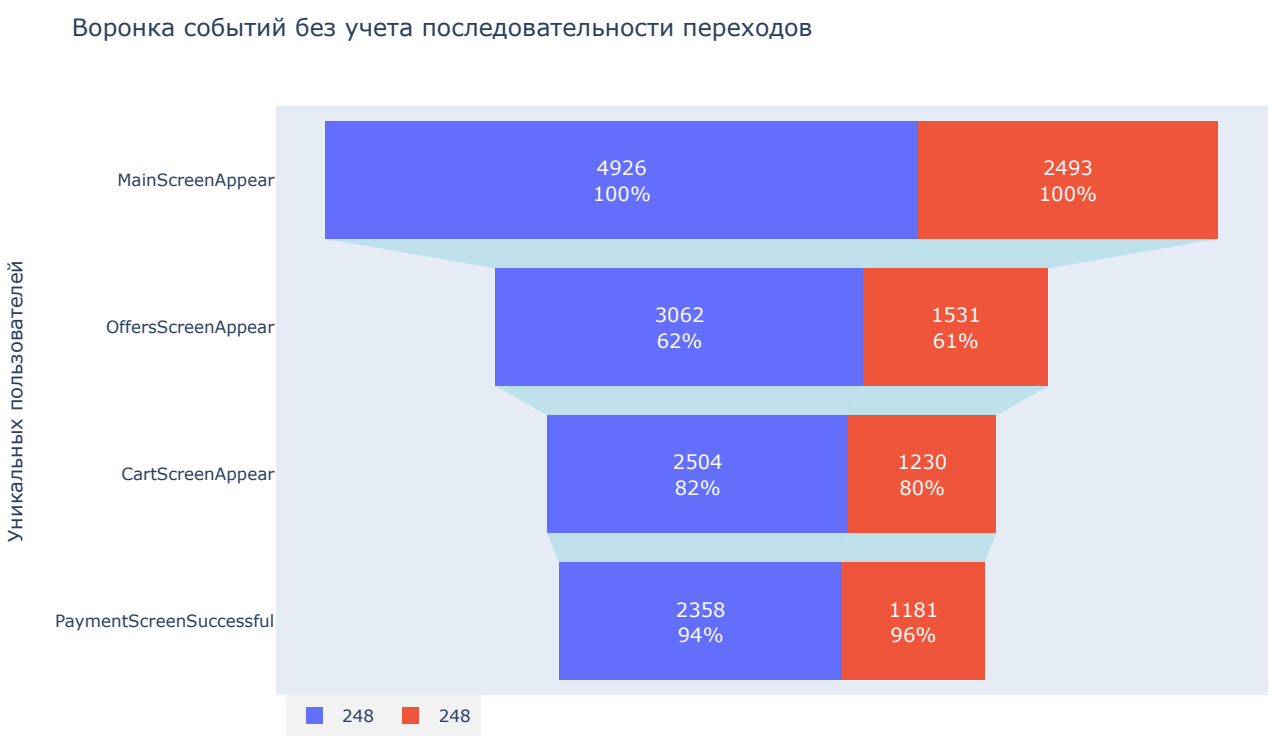
Проверяем статистические критерии разницы между суммой контрольных групп A1 и A2 и группой B.

Считая распределение генеральной совокупности нормальным:

1. Нулевая гипотеза: доли удачных исходов двух групп равны.
2. Альтернативная гипотеза: доли удачных исходов двух групп не равны.

Ввод [39]:
```
z_test('All',248,0.05)
```

```
Группа MainScreenAppear p-value: 0.349
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Группа OffersScreenAppear p-value: 0.446
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Группа CartScreenAppear p-value: 0.187
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Группа PaymentScreenSuccessful p-value: 0.611
Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными
```



Мы провели 16 проверок статистических гипотез с уровнем значимости 0.05 (12 из них проверяли разницу между контрольными группами и группой с изменённым шрифтом) и ни одна из них не выявила значимой разницы.

Если провести эти же исследования при уровне значимости 0.1 (не включенны в итоговую работу) только одна из проверок покажет значимую разницу, между контрольной группой 246 и экспериментальной в доле перехода пользователей в корзину(CartScreenAppear), но эта разница будет не в пользу нашей экспериментальной группы. Т.к. при уровне значимости 0.1 каждый десятый раз можно получать ложный результат, это означает, что изначально выбранный нами уровень значимости 0.05 верен.

Исходя из результатов данного A/A/B-теста, мы можем заключить, что на поведение пользователей изменение шрифта значимого эффекта не оказало. Что можно считать успехом, т.к. целью было узнать не отпугнут ли изменения пользователей. В то же время учитывая результаты эксперимента, если изменение шрифта не продиктовано проблемами в работе приложения, его можно не менять.

популярное событие

Самым часто встречающимся событие во всех группах является MainScreenAppear

Заключение

По результату анализа данных можно сделать следующие выводы:

1. Данные до 01.08.19 признанны не существенными и в анализе не используются. Анализ строится на данных с 01.08.19 по 07.08.19 включительно.

2. Среднее значение числа событий на пользователя составило 32, наиболее часто встречающееся значение - 8

3. Анализ данных в разрезе групп не выявил аномалий.

4. Воронка события взята в виде: MainScreenAppear -> OffersScreenAppear -> CartScreenAppear -> PaymentScreenSuccessful. Событие Tutorial из воронки продаж исключенно.

5. Анализ воронки продаж без учета последовательности шагов показал, что более 60% пользователей, зашедших на сайт доходят до просмотра товаров, 85% из них добавляют товар в корзину и 95% из дообавивших товар в корзину совершает покупку.

   Если же посмотреть на воронку событий с учетом последовательности шагов, то мы видим, что если цифра дошедших до просмотра каталога (около 58%) примерно равна соответствующей цифре в воронке событий без учета последовательности (около 63%), то на следующих шагах цифры намного меньше.

   Это может быть вызванно тем, что имеются другие, кроме предпологаемого нами, пути покупки товара. Например кнопка купить в один клик в катрочке товара.

6. Анализ данных A/A/B теста для контрольных групп A1 и A2 не выявил статистической разницы в данных.

7. Анализ данных A/A/B теста для контрольных групп A1,A2 и их суммы в сравнении с группой B не выявил статистической разницы в данных.

8. Исходя из результатов данного A/A/B-теста, мы можем заключить, что на поведение пользователей изменение шрифта значимого эффекта не оказало.

Ввод [ ]: