**Поведенческая аналитика пользователей мобильного приложения по продаже продуктов питания.**

**Описание проекта.**

Мы работаем в стартапе, который продаёт продукты питания.

Задача:

- разобраться, как ведут себя пользователи нашего мобильного приложения.

Подзадачи:

- необходимо изучить воронку продаж;

- узнать, как пользователи доходят до покупки, сколько пользователей доходит до покупки, а сколько — «застревает» на предыдущих шагах, на каких именно?

- исследовать результаты A/A/B-эксперимента (дизайнеры захотели поменять шрифты во всём приложении, а менеджеры испугались, что пользователям будет непривычно, поэтому договорились принять решение по результатам A/A/B-теста, пользователей разбили на 3 группы: 2 контрольные со старыми шрифтами и одну экспериментальную — с новыми - выясним, какой шрифт лучше).

**Описание данных.**

- EventName — название события;

- DeviceIDHash — уникальный идентификатор пользователя;

- EventTimestamp — время события;

- ExpId — номер эксперимента: 246 и 247 — контрольные группы, а 248 — экспериментальная.

Ввод [1]:

*#подключаем все необходимые библиотеки*

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

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

**from** datetime **import** datetime

**from** scipy **import** stats **as** st

**import** math **as** mth

​

**import** matplotlib.pyplot **as** plt

**%**matplotlib inline

**import** seaborn **as** sns

**import** plotly.express **as** px

**from** plotly **import** graph\_objects **as** go

​

**import** warnings

**Откроем файл с данными и изучим информацию, подготовим данные.**

Ввод [2]:

*#выгрузка и ознакомление с данными датасета*

**try**:

df **=** pd.read\_csv('/datasets/logs\_exp.csv', sep**=**'\t')

**except**:

df **=** pd.read\_csv('https://code.s3.yandex.net/datasets/logs\_exp.csv', sep**=**'\t')

Ввод [3]:

df.head()

Out[3]:

|  | **EventName** | **DeviceIDHash** | **EventTimestamp** | **ExpId** |
| --- | --- | --- | --- | --- |
| **0** | MainScreenAppear | 4575588528974610257 | 1564029816 | 246 |
| **1** | MainScreenAppear | 7416695313311560658 | 1564053102 | 246 |
| **2** | PaymentScreenSuccessful | 3518123091307005509 | 1564054127 | 248 |
| **3** | CartScreenAppear | 3518123091307005509 | 1564054127 | 248 |
| **4** | PaymentScreenSuccessful | 6217807653094995999 | 1564055322 | 248 |

Ввод [4]:

*#изменяю названия столбцов, меняю их вид*

df.columns **=** ['event\_name', 'device\_id', 'event\_time', 'exp\_group']

df.head()

Out[4]:

|  | **event\_name** | **device\_id** | **event\_time** | **exp\_group** |
| --- | --- | --- | --- | --- |
| **0** | MainScreenAppear | 4575588528974610257 | 1564029816 | 246 |
| **1** | MainScreenAppear | 7416695313311560658 | 1564053102 | 246 |
| **2** | PaymentScreenSuccessful | 3518123091307005509 | 1564054127 | 248 |
| **3** | CartScreenAppear | 3518123091307005509 | 1564054127 | 248 |
| **4** | PaymentScreenSuccessful | 6217807653094995999 | 1564055322 | 248 |

Ввод [5]:

*#проверяю данные на наличие пропусков*

df.isna().sum()

Out[5]:

event\_name 0

device\_id 0

event\_time 0

exp\_group 0

dtype: int64

Пропуски в данных отсутствуют.

Ввод [6]:

*#проверяю типы данных*

df.info()

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

RangeIndex: 244126 entries, 0 to 244125

Data columns (total 4 columns):

# Column Non-Null Count Dtype

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

0 event\_name 244126 non-null object

1 device\_id 244126 non-null int64

2 event\_time 244126 non-null int64

3 exp\_group 244126 non-null int64

dtypes: int64(3), object(1)

memory usage: 7.5+ MB

Датафрейм содержит 244126 строк.

Ввод [7]:

*#добавляем столбец с датой и временем с корректным типом данных*

df['date\_time'] **=** [datetime.fromtimestamp(x) **for** x **in** df['event\_time']]

​

*#добавляем столбец с датой*

df['date'] **=** df['date\_time'].dt.normalize()

​

df.head()

Out[7]:

|  | **event\_name** | **device\_id** | **event\_time** | **exp\_group** | **date\_time** | **date** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | MainScreenAppear | 4575588528974610257 | 1564029816 | 246 | 2019-07-25 04:43:36 | 2019-07-25 |
| **1** | MainScreenAppear | 7416695313311560658 | 1564053102 | 246 | 2019-07-25 11:11:42 | 2019-07-25 |
| **2** | PaymentScreenSuccessful | 3518123091307005509 | 1564054127 | 248 | 2019-07-25 11:28:47 | 2019-07-25 |
| **3** | CartScreenAppear | 3518123091307005509 | 1564054127 | 248 | 2019-07-25 11:28:47 | 2019-07-25 |
| **4** | PaymentScreenSuccessful | 6217807653094995999 | 1564055322 | 248 | 2019-07-25 11:48:42 | 2019-07-25 |

Ввод [8]:

*#проверим данные на дубликаты*

df.duplicated().sum()

Out[8]:

413

Ввод [9]:

*#удаление дубликатов*

df **=** df.drop\_duplicates().reset\_index(drop**=True**)

df.duplicated().sum()

Out[9]:

0

Данные не содержат пропусков. Было найдено 413 полных дубликатов - удалены. Сам ДФ содержал всего 244126 строк до удаления дубликатов. Добавлены 2 новых столбца с корректными типами данных. Данные готовы для манипуляций с ними.

**Проверим данные.**

Посчитаем количество событий и пользователей в логе ниже.

Ввод [10]:

print('Всего событий в логе', df['event\_name'].count(),'.')

​

Всего событий в логе 243713 .

Ввод [11]:

print('Всего в логе уникальных уникальных пользователей:', df['device\_id'].nunique (),'.')

​

Всего в логе уникальных уникальных пользователей: 7551 .

Ввод [12]:

*#группируем данные по уникальным пользователям и считаем количество событий для каждого пользователя*

events\_per\_user **=** df.groupby('device\_id')['event\_name'].count()

​

*#вычисляем среднее количество событий на пользователя*

avg\_events\_per\_user **=** events\_per\_user.mean()

​

print('Количество событий в среднем на одного пользователя:', round(avg\_events\_per\_user, 2))

Количество событий в среднем на одного пользователя: 32.28

Ввод [13]:

*#узнаем, за какой период мы обладаем данными*

print('Максимальная дата:', df['date'].max())

print('Минимальная дата:', df['date'].min())

print('Количество дней для анализа:', df['date'].max()**-**df['date'].min())

Максимальная дата: 2019-08-07 00:00:00

Минимальная дата: 2019-07-25 00:00:00

Количество дней для анализа: 13 days 00:00:00

Ввод [14]:

plt.figure(figsize**=**(20, 10))

sns.histplot(x**=**'date\_time', hue**=**'exp\_group', data**=**df)

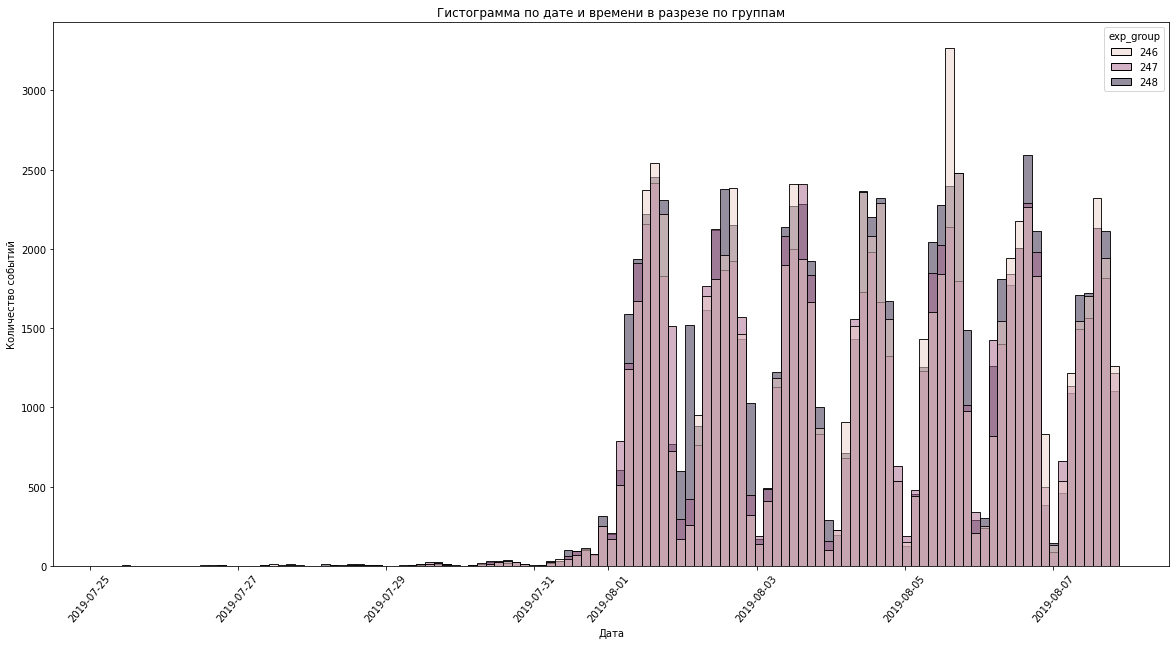
​

plt.ylabel('Количество событий')

plt.xlabel('Дата')

plt.xticks(rotation**=**50)

plt.title('Гистограмма по дате и времени в разрезе по группам');



Изучив график, можно сделать вывод, что наиболее полными данные становятся с 1 августа 2019г, их и оставим для дальнейшего анализа (то есть на самом деле мы располагаем данными для анализа за последнюю неделю из периода). Так же можно сказать, что на графике присутствуют пользователи из всех трех экспериментальных групп. Так же мы можем наблюдать, что среднее количество событий на пользователя действительно около 31т, а пики можно объяснить изменением активности пользователей в течении суток.

Ввод [15]:

*#перезапишем датасет, оставив в нем период с наиболее полными данными*

data **=** df.query('date >= "2019-08-01"')

data.head()

Out[15]:

|  | **event\_name** | **device\_id** | **event\_time** | **exp\_group** | **date\_time** | **date** |
| --- | --- | --- | --- | --- | --- | --- |
| **2826** | Tutorial | 3737462046622621720 | 1564618048 | 246 | 2019-08-01 00:07:28 | 2019-08-01 |
| **2827** | MainScreenAppear | 3737462046622621720 | 1564618080 | 246 | 2019-08-01 00:08:00 | 2019-08-01 |
| **2828** | MainScreenAppear | 3737462046622621720 | 1564618135 | 246 | 2019-08-01 00:08:55 | 2019-08-01 |
| **2829** | OffersScreenAppear | 3737462046622621720 | 1564618138 | 246 | 2019-08-01 00:08:58 | 2019-08-01 |
| **2830** | MainScreenAppear | 1433840883824088890 | 1564618139 | 247 | 2019-08-01 00:08:59 | 2019-08-01 |

Ввод [16]:

*#количество удаленных строк*

df.shape[0] **-** data.shape[0]

Out[16]:

2826

Ввод [17]:

*#доля отброшенных событий*

round((df.shape[0] **-** data.shape[0]) **/** df.shape[0] **\*** 100, 2)

Out[17]:

1.16

Ввод [18]:

*#кол-во отброшенных уникальных пользователей*

df.device\_id.nunique() **-** data.device\_id.nunique()

Out[18]:

17

Ввод [19]:

*#доля отброшенных уникальных пользователей*

round((df.device\_id.nunique() **-** data.device\_id.nunique()) **/** df.device\_id.nunique() **\*** 100, 2)

Out[19]:

0.23

Ввод [20]:

*#поверим есть ли у нас пользователи всех экспериментальных групп после фильтрации*

groups\_after **=** data['exp\_group'].unique()

**if** len(groups\_after) **==** 3:

print('Пользователи из всех трех групп присутствуют после фильтрации.')

**else**:

print('Не все группы присутствуют после фильтрации')

Пользователи из всех трех групп присутствуют после фильтрации.

Ввод [21]:

data.groupby('exp\_group').agg({'device\_id': 'nunique'})

​

Out[21]:

|  | **device\_id** |
| --- | --- |
| **exp\_group** |  |
| **246** | 2484 |
| **247** | 2513 |
| **248** | 2537 |

После фильтрации у нас так же 3 экспериментальные группы: 246 - 2484 пользователя, 247 - 2513 пользователей, 248 - 2537 пользователей.

**Изучим воронку событий.**

Ввод [22]:

*# подсчет частоты событий*

event\_counts **=** data['event\_name'].value\_counts().sort\_values(ascending**=False**)

​

*# вывод результатов*

print('Частота событий:')

print(event\_counts)

Частота событий:

MainScreenAppear 117328

OffersScreenAppear 46333

CartScreenAppear 42303

PaymentScreenSuccessful 33918

Tutorial 1005

Name: event\_name, dtype: int64

В логе есть такие события как MainScreenAppear (Главная страница), OffersScreenAppear (Страница предложения или товара), CartScreenAppear (Страница корзины), PaymentScreenSuccessful (Страница оплаты) и Tutorial (Страница руководства для пользователя). Чаще всего пользователи посещают главную страницу приложения, далее страницу товара, далее страницу корзины и страницу оплаты, меньше всего пользователе интересует руководство по использованию приложения (кто бы сомневался). На первый взгляд логика посещения страниц, исходя из цифр не нарушена.

Ввод [23]:

*# подсчет числа уникальных пользователей для каждого события*

users\_per\_event **=** data.groupby('event\_name')['device\_id'].nunique().sort\_values(ascending**=False**)

​

*# вывод результатов*

print('Число уникальных пользователей для каждого события:')

print(' ')

​

print(users\_per\_event)

print(' ')

​

*# подсчет доли пользователей, которые хоть раз совершали событие*

total\_users **=** data['device\_id'].nunique()

event\_users\_ratio **=** users\_per\_event **/** total\_users

​

*# вывод результатов*

print('Доля пользователей, которые хоть раз совершали событие:')

print(' ')

print(event\_users\_ratio)

Число уникальных пользователей для каждого события:

event\_name

MainScreenAppear 7419

OffersScreenAppear 4593

CartScreenAppear 3734

PaymentScreenSuccessful 3539

Tutorial 840

Name: device\_id, dtype: int64

Доля пользователей, которые хоть раз совершали событие:

event\_name

MainScreenAppear 0.984736

OffersScreenAppear 0.609636

CartScreenAppear 0.495620

PaymentScreenSuccessful 0.469737

Tutorial 0.111495

Name: device\_id, dtype: float64

Как мы видим число пользователей так же равномерно распределено по событиям, логика не нарушена: больше всего посетили главную страницу, меньше всего - страницу руководства. Так же хочу заметить что покупку совершили почти 45% уникальных пользователей приложения - это очень хороший результат( с учетом того, что нормальной считается конверсия в 1,5-2%).

Предполагаю, что события выстраиваются в том порядке, в котором по ним и распределены пользователи по количеству от большего к меньшему: посещение главного экрана, страницы товара, корзина, покупка, наименьшее число пользователей посещает страницу с руководством (не будем учитывать их в дальнейшем), некоторые пользователи не попадают на главную станицу приложения, тк скорее всего переходят сразу на страницу товара по ссылке из письма или соц сетей и тд.

Ввод [24]:

*#по воронке событий посчитаем, какая доля пользователей проходит на следующий шаг воронки*

share\_of\_users **=** data.query('event\_name != "Tutorial"').groupby('event\_name').agg({'device\_id': 'nunique'}) \

.sort\_values(by**=**'device\_id', ascending**=False**).reset\_index()

share\_of\_users['share'] **=** 100 **+** share\_of\_users.device\_id.pct\_change() **\*** 100

share\_of\_users

Out[24]:

|  | **event\_name** | **device\_id** | **share** |
| --- | --- | --- | --- |
| **0** | MainScreenAppear | 7419 | NaN |
| **1** | OffersScreenAppear | 4593 | 61.908613 |
| **2** | CartScreenAppear | 3734 | 81.297627 |
| **3** | PaymentScreenSuccessful | 3539 | 94.777718 |

Ввод [25]:

*#визуализируем воронку событий*

fig **=** go.Figure(

go.Funnel(

orientation **=** "h",

y **=** share\_of\_users.event\_name.values,

x **=** share\_of\_users.device\_id.values,

textposition **=** "inside",

textinfo **=** "value+percent previous"))

fig.update\_layout(title**=**"Воронка шагов пользователей (количество и доля пользователей после предыдущего шага)")

fig.show()

7419100%459362%373481%353995%PaymentScreenSuccessfulCartScreenAppearOffersScreenAppearMainScreenAppear

Воронка шагов пользователей (количество и доля пользователей после предыдущего шага)

Мы можем видеть, что с главной страницы на страницу товара переходят 62% пользователей, к корзине переходят 81% от просматривавших товар, а к покупке 95% от сформировавших корзину. Больше всего пользователей теряется при преходе с главной страницы к карточке товара (или каталогу).

Ввод [26]:

*#посчитаем какая доля пользователей доходит от первого события до оплаты*

payers\_share **=** round(share\_of\_users.device\_id **/** share\_of\_users.device\_id[0] **\*** 100), 2

payers\_share

Out[26]:

(0 100.0

1 62.0

2 50.0

3 48.0

Name: device\_id, dtype: float64,

2)

Доля пользователей оплативший покупку от общего числа пользователей приложения составила - 48% с округлением.

**Изучим результаты эксперимента.**

Ввод [27]:

*#посчитаем количество пользователей в группах*

data.groupby('exp\_group').agg({'device\_id': 'nunique'})

Out[27]:

|  | **device\_id** |
| --- | --- |
| **exp\_group** |  |
| **246** | 2484 |
| **247** | 2513 |
| **248** | 2537 |

У нас так же 3 экспериментальные группы: 246 - 2484 пользователя, 247 - 2513 пользователей, 248 - 2537 пользователей.

Ввод [28]:

events\_users **=** data[data.exp\_group **!=** 248].pivot\_table(index**=**data.event\_name, columns**=**data.exp\_group, aggfunc**=**{'count', 'nunique'})['device\_id']

events\_users.columns **=** ['events\_246', 'events\_247', 'users\_246', 'users\_247']

events\_users['events\_246\_247'] **=** events\_users.events\_246 **+** events\_users.events\_247

events\_users['users\_246\_247'] **=** events\_users.users\_246 **+** events\_users.users\_247

events\_users.sort\_values(by**=**'events\_246', ascending**=False**)

Out[28]:

|  | **events\_246** | **events\_247** | **users\_246** | **users\_247** | **events\_246\_247** | **users\_246\_247** |
| --- | --- | --- | --- | --- | --- | --- |
| **event\_name** |  |  |  |  |  |  |
| **MainScreenAppear** | 37676 | 39090 | 2450 | 2476 | 76766 | 4926 |
| **OffersScreenAppear** | 14767 | 15179 | 1542 | 1520 | 29946 | 3062 |
| **CartScreenAppear** | 14690 | 12434 | 1266 | 1238 | 27124 | 2504 |
| **PaymentScreenSuccessful** | 11852 | 9981 | 1200 | 1158 | 21833 | 2358 |
| **Tutorial** | 317 | 338 | 278 | 283 | 655 | 561 |

Исходя из полученной таблицы можем сделать вывод, что яркие отличия по пользователям между группами не найдены.

Сформулируем гипотезы для всех дальнейших тестов (одинаковые).

Нулевая гипотеза: доли по событиям между группами одинаковы.

Альтернативная гипотеза: доли по событиям между группами различаются.

Ввод [29]:

*#функция для z-теста*

**def** z\_test\_bonferroni(group\_a, group\_b, event, alpha):

group\_users **=** np.array([data.query('event\_name == @event and exp\_group == @group\_a').device\_id.nunique(),

data.query('event\_name == @event and exp\_group == @group\_b').device\_id.nunique()])

total\_users **=** np.array([data.query('exp\_group == @group\_a').device\_id.nunique(),

data.query('exp\_group == @group\_b').device\_id.nunique()])

*# пропорция в первой группе:*

p1 **=** group\_users[0]**/**total\_users[0]

​

*# пропорция во второй группе:*

p2 **=** group\_users[1]**/**total\_users[1]

​

*# пропорция в комбинированном датасете:*

p\_combined **=** (group\_users[0] **+** group\_users[1]) **/** (total\_users[0] **+** total\_users[1])

​

*# 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 **/** total\_users[0] **+** 1 **/** total\_users[1])

)

​

*# задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)*

distr **=** st.norm(0, 1)

p\_value **=** (1 **-** distr.cdf(abs(z\_value))) **\*** 2**/**16 *#добавляем поправку Бонферрони*

​

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 разными'

)

​

Ввод [30]:

*# критический уровень статистической значимости*

alpha **=** 0.05

Ввод [31]:

*#cравнение двух контрольных групп 246 и 247*

group\_a **=** 246

group\_b **=** 247

Ввод [32]:

*# проходим по каждому событию и проводим z-тест*

**for** event **in** df.event\_name.unique():

z\_test\_bonferroni(246, 247, event, 0.05**/**16) *# 0.05 разделили на количество событий*

p-значение: 0.04731623270028812

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.0071604245707136555

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.014302107648748258

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.01550596611576363

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.05860622618285696

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Cамое популярное событие - посещение главной страницы. В контрольных группах это число составило 2 450 и 2 476 пользователей. Нельзя считать отличие между группами статистически достоверным. Разбиение на группы работает корректно.

Ввод [33]:

*#cравнение контрольной группы 246 с экспериментальной*

group\_a **=** 246

group\_b **=** 248

Ввод [34]:

**for** event **in** df.event\_name.unique():

z\_test\_bonferroni(246, 248, event, 0.05**/**16) *# 0.05 разделили на количество событий*

p-значение: 0.01843576208471595

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.013265957973111225

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.004901827023450073

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.013022628376711823

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.05165183756304778

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Мы не можем считать различия статистически значимыми.

Ввод [35]:

*#cравнение контрольной группы 247 с экспериментальной*

group\_a **=** 247

group\_b **=** 248

Ввод [36]:

**for** event **in** df.event\_name.unique():

z\_test\_bonferroni(247, 248, event, 0.05**/**16) *# 0.05 разделили на количество событий*

p-значение: 0.028669085103884467

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.04608384408627478

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.03616373674712364

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.05748636144120163

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.04783274515465631

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Ввод [37]:

*#cравнение результатов объединенной контрольной группы с экспериментальной*

group\_a **=** [246, 247]

group\_b **=** 248

Ввод [38]:

**for** event **in** df.event\_name.unique():

z\_test\_bonferroni([246, 247], 247, event, 0.05**/**16) *# 0.05 разделили на количество событий*

p-значение: 0.05372637311069184

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.022744509733057056

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.030555459106401894

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.03169103101483374

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

p-значение: 0.06026011167320776

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Мы не можем считать различия статистически значимыми ни в одном из сравнений даже с учетом добавленной поправки Бонферрони. Можно сделать вывод, что изменение шрифта не оказывает значительного влияния на конверсию и приложение будет эффективно продавать продукты как со старым так и с новым шрифтом и инвестиции в его редизайн не целесообразны.

Выше я выбрала уровень значимости при проверке статистических гипотез 0.05 (тк при уровне значимости 0.1 в 10% случаев можно ошибочно отклонить нулевую гипотезу при условии, что она верна, растет вероятность ошибки превого рода) и добавила поправку Бонферрони на количество событий - 16 (размер корректировки). Выполнила 4 аа-теста и 12 аб-тестов.

**Вывод.**

В ходе работы над проектом я ознакомилась с данными, изучила их и подготовила для дальнейшего анализа, а именно: получен файл с данными, извлечен, стлбцы приведены к читабильному виду, добавлены новые столбцы с корректными типами данных для чтения даты и времени, данные были проверены на прпуски и дубликаты (полные из последних удалены). Было выявлено, что всего в логе описано 5 уникальных событий, повторение которых составляет в общей сумме 243713 ед, так же в логе 7551 уникальный пользователь, среднее количество событий на пользователя составляет - 32.28. Мы располагаем даннми за 13 дней, наиболее полные данные за последние 7 дней, которые мы и оставилии для АА и АБ-тестов (для групп 246, 247, 248 - контрольная). Далее были изучены воронка событий, результаты эксперимента:

- чаще всего среди событий лога происходит посещение главной страницы;

- с главной страницы на страницу товара переходят 62% пользователей, к корзине переходят 81% от просматривавших товар, а к покупке 95% от сформировавших корзину. Больше всего пользователей теряется при преходе с главной страницы к карточке товара (или каталогу);

- больше всего пользователей теряется при переходе с главной страницы к каталогу - это 38% пользователей;

- в ходе АА и АБ-экперимента между группами, которые видели старый и новый шрифт на страницах приложения мы выяснили что не можем считать различия статистически значимыми (даже с учетом поправки Бонферрони).

Исходя из полученных выше результатов могу рекомендовать не вкладывать средства в изменение шрифта на страницах сайта, тк эксперимент показал не значительное влияние этого параметра на поведение пользователей. Но большая их часть как раз пропадает на моменте прехода с главной страницы в каталог товаров: стоит обратить внимание на внешний вид кнопки Каталога, ее расположение, провести дополнительные исследования визуальных факторов, которые могли бы привлечь больше внимания клиентов к ней. Так же возможно существуют какие-то технические прблемы с кнопкой Каталога - стоит обратить на это внимание тех отдела и провести специальные тесты.