Анализ результатов A/B-тестирования интернет-магазина геймифицированных товаров для спорта и здорового образа жизни

- Автор: Егорова Ольга

Введение

Интернет-магазин по продаже геймифицированных товаров для спорта и здорового образа жизни планирует расширить ассортимент товаров. В связи с жалобами пользователей на сложный интерфейс онлайн-магазина была разработана новая версия сайта и протестирована на части пользователей. Предполагается, что это решение повысит количество пользователей, которые совершат покупку.

Цели и задачи

Цель проекта:

1. оценить корректность проведенного A/B-теста и проанализировать его результаты.

Задачи:

1. Загрузить и подготовить данные к работе.
2. Проверить корректность проведения теста.
3. Оценить результат тестирования

Описание данных

В распоряжении имеется таблица **`ab_test_participants.csv`** с участниками теста.

Поля таблицы:

- `user_id` — идентификатор пользователя;
- `group` — группа пользователя;
- `ab_test` — название теста;
- `device` — устройство, с которого происходила регистрация.

И архив **`ab_test_events.zip`** с одним csv-файлом, в котором собраны события 2020 года;

Поля таблицы:

- `user_id` — идентификатор пользователя;
- `event_dt` — дата и время события;
- `event_name` — тип события;
- `details` — дополнительные данные о событии.

Числовые значения столбца `details`:

- для типа события `registration` (регистрация) в поле указана стоимость привлечения клиента;
- для типа события `purchase` (покупка) в поле указана стоимость покупки.

Текстовые значения `details`:

---	Код	Зона
0	ZONE_CODE00	None
1	ZONE_CODE01	EU
2	ZONE_CODE02	CIS
3	ZONE_CODE03	EU, CIS
4	ZONE_CODE04	APAC
5	ZONE_CODE05	EU, APAC
6	ZONE_CODE06	CIS, APAC
7	ZONE_CODE07	EU, CIS, APAC
8	ZONE_CODE08	N.America
9	ZONE_CODE09	EU, N.America
10	ZONE_CODE10	CIS, N.America
11	ZONE_CODE11	EU, CIS, N.America
12	ZONE_CODE12	APAC, N.America
13	ZONE_CODE13	EU, APAC, N.America
14	ZONE_CODE14	CIS, APAC, N.America
15	ZONE_CODE15	EU, CIS, APAC, N.America

Техническое задание проведённого теста `interface_eu_test`

Проверялось полное обновление дизайна сайта.
Гипотеза: упрощение интерфейса приведёт к тому, что в течение семи дней после регистрации в системе конверсия зарегистрированных пользователей в покупателей увеличится *как минимум на три процентных пункта*.

Параметры теста:

- название теста: `interface_eu_test`;
- группы: A (контрольная), B (новый интерфейс).

Загрузка и знакомство с данными.

Загрузка данных.

Импортируем необходимые для работы библиотеки:

In [1]:
```
#pip install numpy==1.26.4
```

In [2]:
```
#pip install pandas==2.2.2
```

In [3]:
```
#pip install matplotlib==3.9.2
```

In [4]:
```
#pip install scipy==1.13.1
```

In [5]:
```
# Импортируем библиотеки
import io
import zipfile
import requests

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# для статистических тестов
from statsmodels.stats.proportion import proportions_ztest
from scipy.stats import chi2_contingency
from statsmodels.stats.proportion import proportion_effectsize
from statsmodels.stats.power import zt_ind_solve_power
```

In [6]:
```
# Загружаем данные
url1 = 'https://drive.google.com/uc?export=download&id=1n2z0b-OXOUovWOqm9V9IxTG7NgI9TXoR'
url2 = 'https://drive.google.com/uc?export=download&id=1p2qt2E_gGKxYMeOWU9N8GaphDJTeq_cq'

ab_test_participants = pd.read_csv(url1)

# Отправляем GET-запрос к файлу
r = requests.get(url2)

# Создаем файлоподобный объект с помощью io.BytesIO
zip_data = io.BytesIO(r.content)

# Открываем ZIP-архив
z = zipfile.ZipFile(zip_data)

# Получаем список файлов в архиве с помощью z.namelist()
# Берём первый файл [0], знаем, что в архиве ровно один CSV
csv_filename = z.namelist()[0]

# Открываем CSV-файл внутри архива с помощью z.open()
with z.open(csv_filename) as f:
    # Чтение CSV через pandas
    ab_test_events = pd.read_csv(f, parse_dates=['event_dt'], encoding='utf-8',  low_memory=False)
```

In [7]:
```
dict_zones = {
    'code': ['ZONE_CODE00', 'ZONE_CODE01', 'ZONE_CODE02', 'ZONE_CODE03', 'ZONE_CODE04', 'ZONE_CODE05', 'ZONE_CODE06', 'ZONE_CODE07',
          'ZONE_CODE08', 'ZONE_CODE09', 'ZONE_CODE10', 'ZONE_CODE11', 'ZONE_CODE12', 'ZONE_CODE13', 'ZONE_CODE14', 'ZONE_CODE15'],
    'zone': ['', 'EU','CIS', 'EU, CIS',
          'APAC', 'EU, APAC', 'CIS, APAC', 'EU, CIS, APAC',
          'N.America', 'EU, N.America', 'CIS, N.America','EU, CIS, N.America',
          'APAC, N.America', 'EU, APAC, N.America', 'CIS, APAC, N.America', 'EU, CIS, APAC, N.America']
}
zones = pd.DataFrame(dict_zones)
```

Таблица `ab_test_participants`

In [8]:
```
# Выводим основную информацию
ab_test_participants.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14525 entries, 0 to 14524
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   user_id  14525 non-null  object
 1   group    14525 non-null  object
 2   ab_test  14525 non-null  object
 3   device   14525 non-null  object
dtypes: object(4)
memory usage: 454.0+ KB
```

In [9]:
```
ab_test_participants.head()
```

Out[9]:

	user_id	group	ab_test	device
0	0002CE61FF2C4011	B	interface_eu_test	Mac
1	001064FEAAB631A1	B	recommender_system_test	Android
2	001064FEAAB631A1	A	interface_eu_test	Android
3	0010A1C096941592	A	recommender_system_test	Android
4	001E72F50D1C48FA	A	interface_eu_test	Mac

In [10]:
```
# Количество уникальных значений в столбцах
ab_test_participants.nunique()
```

Out[10]:
```
user_id    13638
group          2
ab_test        2
device         4
dtype: int64
```

In [11]:
```
# Выводим уникальные значения столбца group
ab_test_participants['group'].unique()
```

Out[11]:
```
array(['B', 'A'], dtype=object)
```

In [12]:
```
# Выводим уникальные значения столбца ab_test
ab_test_participants['ab_test'].unique()
```

Out[12]:
```
array(['interface_eu_test', 'recommender_system_test'], dtype=object)
```

In [13]:
```
# Выводим уникальные значения столбца device
ab_test_participants['device'].unique()
```

Out[13]:
```
array(['Mac', 'Android', 'iPhone', 'PC'], dtype=object)
```

- Таблица содержит 4 столбца и 14525 строк.
- Пропуски отсутствуют.
- Типы столбцов корректны.
- Количество уникальных пользователей (13638) меньше количества строк таблицы на 887, что говорит о наличии дубликатов. Необходимо проверить данные на полные дубликаты, проверить не пересекаются ли группы для каждого тест по пользователям, проверить не попали ли пользователи с разными типами устройств в одны группу одного теста

Таблица `ab_test_events`

In [14]:
```
# Выводим основную информацию
ab_test_events.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 787286 entries, 0 to 787285
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype         
---  ------      --------------   -----         
 0   user_id     787286 non-null  object        
 1   event_dt    787286 non-null  datetime64[ns]
 2   event_name  787286 non-null  object        
 3   details     249022 non-null  object        
dtypes: datetime64[ns](1), object(3)
memory usage: 24.0+ MB
```

In [15]:
```
# Количество уникальных значений в столбцах
ab_test_events.nunique()
```

Out[15]:
```
user_id       144184
event_dt      606573
event_name         8
details          169
dtype: int64
```

Наша таблица содержит события для двух тестов. Проверим есть ли в данных полные дубликаты. Если такие обнаружатся, удалим их:

In [16]:
```
# Количество полных дубликатов
ab_test_events.duplicated().sum()
```

Out[16]:
```
36318
```

In [17]:
```
# Удалим полные дубликаты
ab_test_events_cleaned = ab_test_events.drop_duplicates()
print('Количество строк:', ab_test_events_cleaned.shape[0])
```

```
Количество строк: 750968
```

In [18]:
```
# Количество пропусков в столбце details
ab_test_events_cleaned['details'].isnull().sum()
```

Out[18]:
```
506063
```

In [19]:
```
# Количество уникальных дат
ab_test_events_cleaned['event_dt'].dt.to_period('D').nunique()
```

Out[19]:
```
31
```

In [20]:
```
# Уникальные даты
ab_test_events_cleaned['event_dt'].dt.to_period('D').unique()
```

Out[20]:
```
<PeriodArray>
['2020-12-01', '2020-12-02', '2020-12-03', '2020-12-04', '2020-12-05',
 '2020-12-06', '2020-12-07', '2020-12-08', '2020-12-09', '2020-12-10',
 '2020-12-11', '2020-12-12', '2020-12-13', '2020-12-14', '2020-12-15',
 '2020-12-16', '2020-12-17', '2020-12-18', '2020-12-19', '2020-12-20',
 '2020-12-21', '2020-12-22', '2020-12-23', '2020-12-24', '2020-12-25',
 '2020-12-26', '2020-12-27', '2020-12-28', '2020-12-29', '2020-12-30',
 '2020-12-31']
Length: 31, dtype: period[D]
```

In [21]:
```
# Выводим уникальные типы событий
ab_test_events_cleaned['event_name'].unique()
```

Out[21]:
```
array(['End of Black Friday Ads Campaign', 'registration', 'product_page',
       'login', 'product_cart', 'purchase',
       'Start of Christmas&New Year Promo',
       'Start of CIS New Year Gift Lottery'], dtype=object)
```

- Первоначальная таблица `ab_test_events` содержала 4 столбца и 787286 строк.

- После удаления полных дубликатов в количестве 36318, новая таблица `ab_test_events_cleaned` содержит 750968 строк.
- Столбец `details` содержит 506063 пропусков.
- Типы столбцов корректны, однако `details` содержит и числовые и строковые данные, что может затруднить обработку данных, например на данном этапе сложно оценить корректность числовых данных этого столбца.
- Таблица содержит данные не за весь 2020 год, а только за декабрь.

Проверка корректности проведения теста `interface_eu_test`

Пользователи участвуют в двух экспериментах с измененным интерфейсом и с новой рекомендательной системой:

In [22]:
```
# Множество пользователей группы B теста interface_eu_test
set_user_b_interfece = set(list(
    ab_test_participants['user_id'][(ab_test_participants['group'] == 'B') & (ab_test_participants['ab_test'] == 'interface_eu_test')]
))

# Множество пользователей группы B теста recommender_system_test
set_user_b_rec_system = set(list(
    ab_test_participants['user_id'][(ab_test_participants['group'] == 'B') & (ab_test_participants['ab_test'] == 'recommender_system_test')]
))

# Множество пользователей группы A теста interface_eu_test
set_user_a_interfece = set(list(
    ab_test_participants['user_id'][(ab_test_participants['group'] == 'A') & (ab_test_participants['ab_test'] == 'interface_eu_test')]
))

# Множество пользователей группы A теста recommender_system_test
set_user_a_rec_system = set(list(
    ab_test_participants['user_id'][(ab_test_participants['group'] == 'A') & (ab_test_participants['ab_test'] == 'recommender_system_test')]
))
```

Пользователи контрольных групп обеих тестов видят интерфейс без изменений. Пользователь может попасть в одну или две выборки без последствий для эксперимента:

In [23]:
```
# Количество пользователей, попавших в две контрольные группы
len(set_user_a_interfece.intersection(set_user_a_rec_system))
```

Out[23]: 325

Для обеспечения чистоты экспериментов, пользователь не может участвовать одновременно в двух тестовых группы обеих экспериментов. То есть он не должен видеть полностью измененный интерфейс сайта и старый интерфейс, но с новой рекомендательной системой. Таких пользователей необходимо исключить:

In [24]:
```
# Количество уникальных пользователей, попавших в группы B обеих экспериментов
len(set_user_b_interfece.intersection(set_user_b_rec_system))
```

Out[24]: 116

Также исключению подлежат пользователи попавшие в контрольную группу одного эксперимента и тестовую другого:

In [25]:
```
# Количество уникальных пользователей, попавших в группы A одного эксперимента и группу B другого
len(set_user_a_interfece.intersection(set_user_b_rec_system))
```

Out[25]: 106

In [26]:
```
# Количество уникальных пользователей, попавших в группы A одного эксперимента и группу B другого
len(set_user_b_interfece.intersection(set_user_a_rec_system))
```

Out[26]: 340

Найдем идентификаторы пользователей, которые могут исказить результаты эксперимента. Для этого объединим множества, результат запишем в `set_users_delete`:

In [27]:
```
# Объединяем множества
set_users_delete = set_user_b_interfece.intersection(set_user_b_rec_system) | \
    set_user_a_interfece.intersection(set_user_b_rec_system) | set_user_b_interfece.intersection(set_user_a_rec_system)
```

In [28]:
```
print('Число пользователей для исключения: ', len(set_users_delete))
```

```
Число пользователей для исключения:  562
```

- Сохраним в `participants_interface` всех участников проведенного теста с названием `interface_eu_test`, исключив пользователей из `set_users_delete`:

In [29]:
```
# Сохраняем данные в новую переменную
ab_test_participants_interface = ab_test_participants[\
    (ab_test_participants['ab_test'] == 'interface_eu_test') & (~ab_test_participants['user_id'].isin(set_users_delete))]
```

In [30]:
```
# Выводим первые 5 строк
ab_test_participants_interface.head()
```

Out[30]:

	user_id	group	ab_test	device
0	0002CE61FF2C4011	B	interface_eu_test	Mac
4	001E72F50D1C48FA	A	interface_eu_test	Mac
5	002412F1EB3F6E38	B	interface_eu_test	Mac
6	002540BE89C930FB	B	interface_eu_test	Android
7	0031F1B5E9FBF708	A	interface_eu_test	Android

In [31]:
```
# Найдем количество строк в таблице
ab_test_participants_interface.shape[0]
```

Out[31]: 10288

- Напомним, что исходная таблица содержит дубликаты по полю `user_id`. Проверим, что в сформированной таблице дубликаты отсутствуют:

In [32]:
```
# Посчитаем количество дубликатов
print('Количество дубликатов: ', ab_test_participants_interface['user_id'].duplicated().sum())
```

```
Количество дубликатов:  0
```

- Проверим равномерность распределения пользователей по группам:

In [33]:
```
ab_test_participants_interface.groupby('group').agg({'user_id':'count'})
```

Out[33]:

group	user_id
A	5277
B	5011

Убедимся, что дисбаланс между группами A и B — это случайность, а не ошибка в распределении. Воспользуемся хи-квадрат тестом:

In [34]:
```
# Уровень значимости
alpha = 0.05

# Таблица сопряженности
contingency_table = ab_test_participants_interface.groupby('group').agg({'user_id':'count'})

# Хи-квадрат тест на независимость
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Выводим результат
if p_value > alpha:
    print(f'p-value={p_value:.4f} > {alpha}')
    print('Распределение пользователей между группами корректно. Дисбаланс между группами - случайность!')
else:
    print(f'p-value={p_value:.4f} < {alpha}')
    print('⚠Есть статистически значимый дисбаланс!')
```

```
p-value=1.0000 > 0.05
Распределение пользователей между группами корректно. Дисбаланс между группами - случайность!
```

- Проверим равномерность распределения пользователей по группам и типам устройств:

In [35]:
```
ab_test_participants_interface.groupby(['group', 'device'])['user_id'].count().unstack(fill_value=0)
```

Out[35]:

group	Android	Mac	PC	iPhone
A	2400	559	1319	999
B	2218	511	1302	980

Убедимся, что дисбаланс между группами A и B — это случайность, а не ошибка в распределении. Воспользуемся хи-квадрат тестом:

In [36]:
```
# Уровень значимости
alpha = 0.05

# Таблица сопряженности
contingency_table = ab_test_participants_interface.groupby(['group', 'device'])['user_id'].count().unstack(fill_value=0)
# Хи-квадрат тест на независимость
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Выводим результат
if p_value > alpha:
    print(f'p-value={p_value:.4f} > {alpha}')
    print('Распределение пользователей между группами корректно. Дисбаланс между группами - случайность!')
else:
    print(f'p-value={p_value:.4f} < {alpha}')
    print('⚠Есть статистически значимый дисбаланс!')
```

```
p-value=0.4330 > 0.05
Распределение пользователей между группами корректно. Дисбаланс между группами - случайность!
```

Пользователи корректно распределены по тестовым группам: группы сопоставимы по размеру и не пересекаются, пользователи внутри групп не повторяются и равномерно распределены по типам устройств

Анализ данных активности пользователей, участвующих в тесте `interface_eu_test`

Подготовим данные датафрейма `ab_test_events_cleaned`

In [37]:
```
ab_test_events_cleaned.head()
```

Out[37]:

	user_id	event_dt	event_name	details
0	GLOBAL	2020-12-01 00:00:00	End of Black Friday Ads Campaign	ZONE_CODE15
1	CCBE9E7E99F94A08	2020-12-01 00:00:11	registration	0.0
2	GLOBAL	2020-12-01 00:00:25	product_page	NaN
3	CCBE9E7E99F94A08	2020-12-01 00:00:33	login	NaN
4	CCBE9E7E99F94A08	2020-12-01 00:00:52	product_page	NaN

- Для дальнейшей работы нам потребуются записи только о регистрации и покупках пользователей. Сохраним в отдельный датафрейм только те строки, для которых в поле `event_name` указано событие `registration` или `purchase`:

In [38]:
```
# Сохраним в df_events все записи о регистрациях и покупках
df_events = ab_test_events_cleaned[(ab_test_events_cleaned['event_name'] == 'registration') | (ab_test_events_cleaned['event_name'] == 'purchase')]
```

- Проверим корректность данных. Пользователь может иметь несколько записей о покупках, но запись о регистрации очевидно должна быть одна. Не будем исключать возможность технических сбоев и проверим, что в таблице нет пользователей с несколькими регистрациями:

In [39]:
```
# Группируем по пользователям и считаем количество регистраций, выводим пользователей с количеством регистрации более 1
df_events[df_events['event_name'] == 'registration'].groupby('user_id').agg({'event_name':'count'}).query('event_name > 1')
```

Out[39]:

user_id	event_name

- Оставим в таблице `df_events` только записи пользователей, участвующих только в тесте `interface_eu_test` и имеющие оба типа событий, то есть и регистрацию и оплату:

In [40]:
```
# Получаем список пользователей теста interface_eu_test
list_user_test = ab_test_participants_interface['user_id'].unique().tolist()
print('Количество уникальных пользователей теста interface_eu_test:', len(list_user_test))
```

```
Количество уникальных пользователей теста interface_eu_test: 10288
```

In [41]:
```
# Получаем список пользователей, у которых есть ровно 2 типа событий
list_user_2_type = df_events[['user_id', 'event_name']].groupby('user_id').agg({'event_name':'nunique'}).query('event_name == 2').index.tolist()

print('Количество уникальных пользователей с регистрацией и одной или более оплатой:', len(list_user_2_type))
```

```
Количество уникальных пользователей с регистрацией и одной или более оплатой: 44758
```

In [42]:
```
# Получаем общие элементы двух множеств
list_user = list(set(list_user_test) & set(list_user_2_type))
print('Количество уникальных пользователей теста interface_eu_test с регистрацией и одной или более оплатой:', len(list_user))
```

```
Количество уникальных пользователей теста interface_eu_test с регистрацией и одной или более оплатой: 3518
```

Из датафрейма df_events выбираем пользователей из полученного списка list_users:

In [43]:
```
# Фильтруем записи на основе списка пользователей list_user
df_events = df_events[df_events['user_id'].isin(list_user)]
df_events
```

Out[43]:

	user_id	event_dt	event_name	details
69718	C64851EFACDDDFEB	2020-12-06 22:30:07	registration	-2.38
70291	DD3B49B4AF10F101	2020-12-06 23:32:12	registration	0.0
70728	FBC41B97316A7769	2020-12-07 00:06:57	registration	-5.22
70832	F1995B461E27CB8D	2020-12-07 00:11:09	registration	-1.42
70875	373C038EF663BDF5	2020-12-07 00:12:38	registration	0.0
...	...	...	...	...
774698	5C31C0607EFC3C98	2020-12-30 02:13:55	purchase	4.29
776685	4584E51B99DE51AE	2020-12-30 07:48:27	purchase	8.99
777473	F80C9BDDEA02E53C	2020-12-30 10:01:32	purchase	4.49
777479	F80C9BDDEA02E53C	2020-12-30 10:02:43	purchase	4.49
777488	F80C9BDDEA02E53C	2020-12-30 10:03:51	purchase	4.49

12626 rows × 4 columns

- В гипотезе указаны временные рамки "в течение 7 дней", поэтому:

а) для каждого пользователя необходимо рассчитать время между регистрацией и совершением первой покупки;
б) оставим пользователей, для которых время между регистрацией и покупкой составило не более 7 дней.

Для удобства сгруппируем датафрейм `df_events` по полям `user_id` и `event_name` и для поля `event_dt` возвращаем минимальную дату. С помощью функции `.unstack()` переместим данные из внутреннего уровня индекса в заголовки столбцов. В результате в столбце `registration` будет дата регистрации, а в столбце `purchase` - минимальная (=первая) дата оплаты.

In [44]:
```
# Группируем по пользователю и находим минимальную дату для каждого из событий
df_events = df_events.groupby(['user_id', 'event_name'])['event_dt'].min().unstack(fill_value=0)
```

In [45]:
```
# Сбрасываем индексы
df_events = df_events.reset_index()
df_events
```

Out[45]:

	event_name	user_id	purchase	registration
0		0031F1B5E9FBF708	2020-12-15 18:54:51	2020-12-14 00:47:10
1		004C58ADE7CA8C4A	2020-12-20 01:13:34	2020-12-18 15:14:53
2		0050F43F34C955F4	2020-12-13 19:44:29	2020-12-13 19:41:56
3		0053DD654C9513D6	2020-12-15 08:46:21	2020-12-12 04:35:20
4		0082295A41A867B5	2020-12-21 17:21:24	2020-12-16 07:07:52
...		...	...	...
3513		FF827554725859E2	2020-12-28 18:57:32	2020-12-23 03:50:20
3514		FFA72985E689ABBB	2020-12-18 19:16:48	2020-12-15 20:18:10
3515		FFD58017F5FA2DAC	2020-12-16 22:57:35	2020-12-13 00:55:02
3516		FFE40BDB7364E966	2020-12-24 18:57:56	2020-12-22 04:07:47
3517		FFE7FC140521F5F6	2020-12-26 14:37:21	2020-12-23 09:10:16

3518 rows × 3 columns

Создадим новое поле `delta` с разницей между датой регистрации и датой оплаты:

In [46]:
```
# Создаем столбец
df_events['delta'] = df_events['purchase'] - df_events['registration']
```

Поскольку нас интересуют первые 7 дней после регистрации, ограничим значения столбца `delta` и сохраним результат в `df_events_7_days`:

In [47]:
```
# Фильтруем данные
df_events_7_days = df_events[df_events['delta'] < pd.to_timedelta('7 day')]
```

In [48]:
```
df_events_7_days.head()
```

Out[48]:

	event_name	user_id	purchase	registration	delta
0		0031F1B5E9FBF708	2020-12-15 18:54:51	2020-12-14 00:47:10	1 days 18:07:41
1		004C58ADE7CA8C4A	2020-12-20 01:13:34	2020-12-18 15:14:53	1 days 09:58:41
2		0050F43F34C955F4	2020-12-13 19:44:29	2020-12-13 19:41:56	0 days 00:02:33
3		0053DD654C9513D6	2020-12-15 08:46:21	2020-12-12 04:35:20	3 days 04:11:01
4		0082295A41A867B5	2020-12-21 17:21:24	2020-12-16 07:07:52	5 days 10:13:32

Датафрейм `df_events_7_days` готов к дальнейшей работе.

In [49]:
```
df_events_7_days.shape[0]
```

Out[49]: 2934

Объединение таблиц `ab_test_participants_interface` и `df_event`

- Объединим таблицу `ab_test_participants_interface` и таблицу `df_events_7_days` по полю `user_id`. Напомним, что таблица `df_events_7_days` содержит данные только по пользователям `interface_eu_test` и только по тем пользователям, которые и зарегистрировались и совершили покупку:

In [50]:
```
# Объединяем таблицы
df_test_interface = df_events_7_days.merge(ab_test_participants_interface, on='user_id', how='left')
df_test_interface.head()
```

Out[50]:

	user_id	purchase	registration	delta	group	ab_test	device
0	0031F1B5E9FBF708	2020-12-15 18:54:51	2020-12-14 00:47:10	1 days 18:07:41	A	interface_eu_test	Android
1	004C58ADE7CA8C4A	2020-12-20 01:13:34	2020-12-18 15:14:53	1 days 09:58:41	B	interface_eu_test	Android
2	0050F43F34C955F4	2020-12-13 19:44:29	2020-12-13 19:41:56	0 days 00:02:33	A	interface_eu_test	PC
3	0053DD654C9513D6	2020-12-15 08:46:21	2020-12-12 04:35:20	3 days 04:11:01	B	interface_eu_test	PC
4	0082295A41A867B5	2020-12-21 17:21:24	2020-12-16 07:07:52	5 days 10:13:32	A	interface_eu_test	iPhone

- Удалим столбец `ab_test` с названием теста `interface_eu_test`, он далее не понадобится:

In [51]:
```
# Удалим столбец с названием теста
df_test_interface = df_test_interface.drop('ab_test', axis=1)
```

- Рассчитаем необходимый размер выборки для получения статистически значимых результатов A/B теста.

Заданные параметры:
-Достоверность теста 95% — (α = 0.05).
-Мощность теста — 80% (0.8)
-Базовый показатель конверсии — 30%

Согласно гипотезе, конверсия после упрощения интерфейса изменится на 3 процентных пункта, значит ожидаемый показатель конверсии будет 30% + 3% = 33%.

Рассчитаем размер эффекта для пропорций:

In [52]:
```
# Базовый показатель конверсии
p1 = 0.3
# Ожидаемый показатель конверсии
p2 = p1 + 0.03
```

```
# Считаем размер эффекта
effect_size = proportion_effectsize(p1, p2)
print(f'Размер эффекта: {abs(effect_size):.4f}')
```

```
Размер эффекта: 0.0646
```

Размер эффекта очень мал, потребуется большая выборка для обнаружения эффекта.

Рассчитаем размер выборки:

In [53]:
```
# Уровень значимости
alpha = 0.05
# Мощность
power = 0.8

# Считаем размер выборки
sample_size = zt_ind_solve_power(
    effect_size=abs(effect_size),
    alpha=alpha,
    power=power,
    ratio=1,
    alternative='larger'
)
print(f"Необходимый размер выборки для каждой группы: {round(sample_size)}")
```

```
Необходимый размер выборки для каждой группы: 2963
```

Для обнаружения роста конверсии с 30% до 33% с заданной мощностью 80% и уровнем значимости 5% потребуется по 2963 пользователя для каждой группы.

- Рассчитаем для каждой группы количество посетителей, совершивших покупку, и общее количество посетителей:

In [54]:
```
print('Количество уникальных пользователей:')
print('- тестовых групп: ', ab_test_participants_interface['user_id'].nunique())
print('- группы A: ', ab_test_participants_interface['user_id'][ab_test_participants_interface['group'] == 'A'].nunique())
print('- группы B: ', ab_test_participants_interface['user_id'][ab_test_participants_interface['group'] == 'B'].nunique())
```

```
Количество уникальных пользователей:
- тестовых групп:  10288
- группы A:  5277
- группы B:  5011
```

In [55]:
```
print('Количество уникальных пользователей совершивших покупку в течение 7 дней после регистрации:')
print('- в двух группах: ', df_test_interface['user_id'].nunique())
print('- группы A: ', df_test_interface['user_id'][df_test_interface['group'] == 'A'].nunique())
print('- группы B: ', df_test_interface['user_id'][df_test_interface['group'] == 'B'].nunique())
```

```
Количество уникальных пользователей совершивших покупку в течение 7 дней после регистрации:
- в двух группах:  2934
- группы A:  1454
- группы B:  1480
```

In [56]:
```
# Считаем общее число пользователей в каждой группе
visitors = ab_test_participants_interface.groupby('group').agg({'user_id':'nunique'})
visitors.columns = ['visitor']
#
shoppers = df_test_interface.groupby('group').agg({'user_id':'nunique'})
shoppers.columns = ['shopper']

# Объединим две таблицы
visitors_and_shoppers = visitors.join(shoppers, how='inner')
visitors_and_shoppers
```

Out[56]:

group	visitor	shopper
A	5277	1454
B	5011	1480

Размер групп больше запланированного объема в ~ 1.7 раза. Мощность теста увеличилась, что даст возможность обнаружить меньший эффект.

In [57]:
```
# Добавим столбец с долей
visitors_and_shoppers['share'] = round(visitors_and_shoppers['shopper'] * 100 / visitors_and_shoppers['visitor'] , 2)
visitors_and_shoppers
```

Out[57]:

group	visitor	shopper	share
A	5277	1454	27.55
B	5011	1480	29.54

In [58]:
```
shopper_a = visitors_and_shoppers.loc['A', 'share']
shopper_b = visitors_and_shoppers.loc['B', 'share']

# Считаем процентную разницу
percentage_difference = 100 * abs(shopper_a - shopper_b) / shopper_a
print(f'Процентная разница в конверсии групп A и B: {round(percentage_difference,2)}%')
```

```
Процентная разница в конверсии групп A и B: 7.22%
```

Конверсия в контрольной группе составляет 27.55%, что на 2,45% ниже базового показателя конверсии. При этом пользователи группы B показывают большую активность, чем пользователи из группы A. Разница между ними составляет чуть меньше 2-х процентов - 1.99%, что в относительном выражении соответствует увеличению конверсии на 7.22%. Можно предположить, что действительно полное изменение дизайна сайта положительного влияния. Однако является ли данный показатель статистически значимым покажет тест.

Проведите оценку результатов A/B-тестирования:

Выше уже обсуждался вопрос, что пользователи групп A и B не пересекаются, пользователи внутри групп не повторяются и равномерно распределены по типам устройств, а группы сопоставимы по размеру.

Проверим, выполняется ли предпосылка использования z-теста пропорций о достаточном размере выборок.
Пусть {nA,nB} — это размеры выборок группы A и B, то есть количество посетителей из группы A и B соответственно,
{mA,mB} — это количество успехов в выборках A и B, то есть количество пользователей из группы A и B, совершивших покупку,
{pA,pB} - это вероятности успехов для групп A и B.

In [59]:
```
# Размеры выборок
n_a = visitors_and_shoppers.loc['A', 'visitor']
n_b = visitors_and_shoppers.loc['B', 'visitor']
# Количество успехов
m_a = visitors_and_shoppers.loc['A', 'share']
m_b = visitors_and_shoppers.loc['B', 'share']
# Доли успехов
p_a = m_a/n_a
p_b = m_b/n_b

# Проверяем условие и выводим результат
if (p_a*n_a > 10) and ((1-p_a)*n_a > 10) and (p_b*n_b > 10) and ((1-p_b)*n_b > 10):
    print('Предпосылка о достаточном количестве данных выполняется!')
else:
    print('Предпосылка о достаточном количестве данных НЕ выполняется!')
```

```
Предпосылка о достаточном количестве данных выполняется!
```

Сформулируем гипотезы для Z-теста пропорций:
• H0: pA >= pB , то есть конверсия зарегистрированных пользователей в покупателей не увеличится в группе с измененным интерфейсом.
• H1: pA < pB, то есть конверсия зарегистрированных пользователей в покупателей увеличится в группе с измененным интерфейсом.

In [60]:
```
alpha = 0.05

stat_ztest, p_value_ztest = proportions_ztest(
    [m_a, m_b],
    [n_a, n_b],
    alternative='larger'
)

if p_value_ztest > alpha:
    print(f'pvalue={p_value_ztest} > {alpha}')
    print('Нулевая гипотеза находит подтверждение! Конверсия пользователей в покупатели не увеличится с изменением интерфейса.')
else:
    print(f'pvalue={p_value_ztest} < {alpha}')
    print('Нулевая гипотеза не находит подтверждения!')
```

```
pvalue=0.6773006813463973 > 0.05
Нулевая гипотеза находит подтверждение! Конверсия пользователей в покупатели не увеличится с изменением интерфейса.
```

Рассчитаем фактическую мощность теста:

In [61]:
```
# Считаем фактическую мошность
power = zt_ind_solve_power(
    effect_size=effect_size,
    nobs1=min(n_a, n_b),
    alpha=alpha,
    ratio=1
)
print(f"Мощность теста: {power:.2f}")
```

```
Мощность теста: 0.90
```

Больший объем выборок изменил мощность с 80% до 90%, что увеличило чувствительность теста и позволяет обнаружить даже небольшие изменения. В результате проведенного z-теста пропорций получаем p_value_ztest = 0.6773, то есть вероятность наблюдать разницу в конверсиях (или еще более экстремальную) при условии, что нулевая гипотеза верна, составляет 67.73%. Поскольку p_value_ztest больше уровня значимости, нет статистически значимых оснований отвергнуть нулевую гипотезу, то есть наблюдаемая разница в конверсиях может быть случайностью и новый дизайн не привел к улучшению по сравнению со старым.
На основании имеющихся данных изменение дизайна не дало положительного эффекта, дальнейшее внедрение нового дизайна не целесообразно и может привести к дополнительным тратам без увеличения метрик. Однако стоит провести дополнительное исследование и проверить статистическую значимость для отдельных типов устройств.
Согласно нашим ожиданиям, измененный дизайн должен был увеличить конверсию на 3 процентных пункта. Однако тест показал, что если изменение конверсии и есть, то оно настолько мало, что даже с высокой мощностью его не удалось зафиксировать. Таким образом, улучшения конверсии на 3% мы не достигли.
