A/B-тестирование

Цель исследования: провести оценку результатов A/B-теста, между группой с новой системой рекомендаций(B) и контрольной группой со старой(A), группа B будет признана успешной если конверсия в каждой метрикие(просмотр товара, просмотр корзины, покупка) улучшится не менее, чем на 10%.

Исследуем данные

импортируем необходимые бибилотеки

```
import pandas as pd
import datetime as dt
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import math as mth
from plotly import graph_objects as go
from scipy import stats as st
```

чтение файлов с данными и сохранение их в переменные

```
final_ab_events = pd.read_csv('/datasets/final_ab_events.csv')

ab_project_marketing_events =
pd.read_csv('/datasets/ab_project_marketing_events.csv')

final_ab_new_users = pd.read_csv('/datasets/final_ab_new_users.csv')

final_ab_participants =
pd.read_csv('/datasets/final_ab_participants.csv')
```

рассмотрим их

```
pd.set_option('display.max_columns', None)
for elem in [final_ab_events, ab_project_marketing_events,
final_ab_new_users, final_ab_participants]:
    print(elem.head(10))
```

```
            user_id             event_dt event_name  details
0  E1BDDCE0DAFA2679  2020-12-07 20:22:03   purchase    99.99
1  7B6452F081F49504  2020-12-07 09:22:53   purchase     9.99
2  9CD9F34546DF254C  2020-12-07 12:59:29   purchase     4.99
3  96F27A054B191457  2020-12-07 04:02:40   purchase     4.99
4  1FD7660FDF94CA1F  2020-12-07 10:15:09   purchase     4.99
```

```
5  831887FE7F2D6CBA  2020-12-07 06:50:29   purchase     4.99
6  6B2F726BFD5F8220  2020-12-07 11:27:42   purchase     4.99
7  BEB37715AACF53B0  2020-12-07 04:26:15   purchase     4.99
8  B5FA27F582227197  2020-12-07 01:46:37   purchase     4.99
9  A92195E3CFB83DBD  2020-12-07 00:32:07   purchase     4.99
                              name                    regions
start_dt  \
0          Christmas&New Year Promo             EU, N.America  2020-
12-25
1      St. Valentine's Day Giveaway  EU, CIS, APAC, N.America  2020-
02-14
2            St. Patric's Day Promo             EU, N.America  2020-
03-17
3                      Easter Promo  EU, CIS, APAC, N.America  2020-
04-12
4                  4th of July Promo                 N.America  2020-
07-04
5         Black Friday Ads Campaign  EU, CIS, APAC, N.America  2020-
11-26
6            Chinese New Year Promo                    APAC  2020-
01-25
7  Labor day (May 1st) Ads Campaign             EU, CIS, APAC  2020-
05-01
8   International Women's Day Promo             EU, CIS, APAC  2020-
03-08
9   Victory Day CIS (May 9th) Event                      CIS  2020-
05-09

    finish_dt
0  2021-01-03
1  2020-02-16
2  2020-03-19
3  2020-04-19
4  2020-07-11
5  2020-12-01
6  2020-02-07
7  2020-05-03
8  2020-03-10
9  2020-05-11
            user_id  first_date     region   device
0  D72A72121175D8BE  2020-12-07         EU       PC
1  F1C668619DFE6E65  2020-12-07  N.America  Android
2  2E1BF1D4C37EA01F  2020-12-07         EU       PC
3  50734A22C0C63768  2020-12-07         EU   iPhone
4  E1BDDCE0DAFA2679  2020-12-07  N.America   iPhone
5  137119F5A9E69421  2020-12-07  N.America   iPhone
6  62F0C741CC42D0CC  2020-12-07       APAC   iPhone
7  8942E64218C9A1ED  2020-12-07         EU       PC
8  499AFACF904BBAE3  2020-12-07  N.America   iPhone
```

```
9  FFCEA1179C253104  2020-12-07         EU  Android
           user_id group                  ab_test
0  D1ABA3E2887B6A73     A  recommender_system_test
1  A7A3664BD6242119     A  recommender_system_test
2  DABC14FDDFADD29E     A  recommender_system_test
3  04988C5DF189632E     A  recommender_system_test
4  482F14783456D21B     B  recommender_system_test
5  4FF2998A348C484F     A  recommender_system_test
6  7473E0943673C09E     A  recommender_system_test
7  C46FE336D240A054     A  recommender_system_test
8  92CB588012C10D3D     A  recommender_system_test
9  057AB296296C7FC0     B  recommender_system_test
```

требуется ли преобразование типов?

```
for elem in [final_ab_events, ab_project_marketing_events,
final_ab_new_users, final_ab_participants]:
    print(elem.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440317 entries, 0 to 440316
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   user_id     440317 non-null  object 
 1   event_dt    440317 non-null  object 
 2   event_name  440317 non-null  object 
 3   details     62740 non-null   float64
dtypes: float64(1), object(3)
memory usage: 13.4+ MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   name       14 non-null     object
 1   regions    14 non-null     object
 2   start_dt   14 non-null     object
 3   finish_dt  14 non-null     object
dtypes: object(4)
memory usage: 576.0+ bytes
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61733 entries, 0 to 61732
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
```

```
 0   user_id     61733 non-null  object
 1   first_date  61733 non-null  object
 2   region      61733 non-null  object
 3   device      61733 non-null  object
dtypes: object(4)
memory usage: 1.9+ MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18268 entries, 0 to 18267
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   user_id  18268 non-null  object
 1   group    18268 non-null  object
 2   ab_test  18268 non-null  object
dtypes: object(3)
memory usage: 428.3+ KB
None
```

приведем столбцы со временем к нужным типам (event_dt, start_dt, finish_dt, first_date)

```
final_ab_events['event_dt'] = final_ab_events['event_dt'].map(
   lambda x: dt.datetime.strptime(x, '%Y-%m-%d %H:%M:%S'))

ab_project_marketing_events['start_dt'] =
ab_project_marketing_events['start_dt'].map(
   lambda x: dt.datetime.strptime(x, '%Y-%m-%d'))

ab_project_marketing_events['finish_dt'] =
ab_project_marketing_events['finish_dt'].map(
   lambda x: dt.datetime.strptime(x, '%Y-%m-%d'))

final_ab_new_users['first_date'] =
final_ab_new_users['first_date'].map(
   lambda x: dt.datetime.strptime(x, '%Y-%m-%d'))
```

опишем природу пропущенных значений и дубликатов, если нам удастся их обнаружить

судя по результатам функции info, пропуски есть только в столбце details первого датафрейма(final_ab_events)

```
print(final_ab_events.groupby('event_name').count())
```

```
              user_id  event_dt  details
event_name
login          189552    189552        0
product_cart    62462     62462        0
product_page   125563    125563        0
purchase        62740     62740    62740
```

не будем ничего менять, пропуски появляются в ивентах (product_cart, product_page), детали которых не могут исчисляться цифрами

Проверим корректность всех пунктов технического задания

отфильтруем в отдлеьный датафрейм всех пользоватлей из нужного нам теста

```
needed_user = []
for elem in
(final_ab_participants['user_id'].loc[(final_ab_participants['ab_test'
] == 'recommender_system_test') | ((final_ab_participants['ab_test']
== 'interface_eu_test') & (final_ab_participants['group'] == 'A'))]):
   needed_user.append(elem)

useless_user = []
for elem in (final_ab_participants['user_id'].loc[
   (final_ab_participants['ab_test'] == 'interface_eu_test') &
(final_ab_participants['group'] == 'B')]):
   useless_user.append(elem)
```

пользователи только из нужного теста(рекомендации)

заранее выделим пользователей групп А и Б теста с новой системой рекомендаций

```
needed_user_A = []
for elem in
final_ab_participants['user_id'].loc[final_ab_participants['group'] !=
'B']:
   needed_user_A.append(elem)

needed_user_B = []
for elem in (final_ab_participants['user_id'].loc[
   (final_ab_participants['ab_test'] == 'recommender_system_test') &
(final_ab_participants['group'] != 'A')]):
   needed_user_B.append(elem)

print('подходящие пользователи(A/B recommender_system_test / A
interface_eu_test):', len(set(needed_user)))
```

```
подходящие пользователи(A/B recommender_system_test / A
interface_eu_test): 11713
```

Всего пользователей до удаления

```
final_ab_participants.groupby(['ab_test', 'group'])['user_id'].count()
```

```
ab_test                 group
interface_eu_test       A        5831
                        B        5736
recommender_system_test A        3824
```

```
                          B        2877
Name: user_id, dtype: int64
```

```
print('неподходящие пользователи(B interface_eu_test):',
len(set(useless_user)))
```

```
неподходящие пользователи(B interface_eu_test): 5736
```

```
intersect_users = np.intersect1d(useless_user, needed_user)
print('всего пересекающихся пользователей между разными
тестами:',len(set(intersect_users)))
```

```
всего пересекающихся пользователей между разными тестами: 783
```

```
intersect_users_groups = np.intersect1d(needed_user_B, needed_user_A)
print('всего пересекающихся пользователей между группами теста с
новыми рекомендациями:',len(set(intersect_users_groups)))
```

```
всего пересекающихся пользователей между группами теста с новыми
рекомендациями: 337
```

внутри теста пересечений нет

уберем участвующих одновременно в разных тестах

уберем участвующих в двух группах одного теста одновременно

```
final_ab_participants_filtered = final_ab_participants[
   (final_ab_participants.user_id.isin(intersect_users) == False) & (
           final_ab_participants.user_id.isin(intersect_users_groups)
== False)]
```

Уберем группу B другого теста, оставим всеъ пользователей контрольной группы A обоих тестов.

```
final_ab_participants_filtered =
final_ab_participants_filtered.loc[(final_ab_participants_filtered['gr
oup'] != 'B') | (final_ab_participants_filtered['ab_test'] !=
'interface_eu_test')]
```

Пользователи после удаления

```
final_ab_participants_filtered.groupby(['ab_test', 'group']).count()
```

```
                              user_id
ab_test                group
interface_eu_test      A         5494
recommender_system_test A        3385
                       B         2196
```

```
final_ab_participants_filtered =
final_ab_participants_filtered.query('ab_test ==
"recommender_system_test"')
```

Отфильтруем только европейских пользователей

```
final_ab_new_users_EU = final_ab_new_users.query('region == "EU"')
['user_id'].tolist()
```

Согласно ТЗ уберем неевропейских пользователей

```
print('всего неевропейских пользователей:',
len(final_ab_participants_filtered.query('user_id not in
@final_ab_new_users_EU')))
```

```
всего неевропейских пользователей: 350
```

```
final_ab_participants_filtered =
final_ab_participants_filtered.query('user_id in
@final_ab_new_users_EU')

print('всего уникальных пользователей после удаления',
len(final_ab_participants_filtered['user_id'].unique()))
```

```
всего уникальных пользователей после удаления 5231
```

Оценим корректность проведения теста

сответсвие данных требованиям технического задания группы: A — контрольная, B — новая платёжная воронка - ВЕРНО, хоть есть и еще один тест

```
print(final_ab_participants.groupby(['group', 'ab_test'])
['group'].count())
```

```
group  ab_test
A      interface_eu_test           5831
       recommender_system_test     3824
B      interface_eu_test           5736
       recommender_system_test     2877
Name: group, dtype: int64
```

дата запуска: 2020-12-07 - ВЕРНО дата остановки набора новых пользователей: 2020-12-21 - НЕ ВЕРНО

```
needed_user_filter = []
for elem in (
       final_ab_participants_filtered['user_id']):
```

```
   needed_user_filter.append(elem)

print('дата запуска набора:', 'ожидаемая 2020-12-07', 'реальная',

final_ab_new_users[final_ab_new_users.user_id.isin(needed_user_filter)
== True]['first_date'].min())
print('дата остановки набора:', 'ожидаемая 2020-12-21', 'реальная',

final_ab_new_users[final_ab_new_users.user_id.isin(needed_user_filter)
== True]['first_date'].max())
```

```
дата запуска набора: ожидаемая 2020-12-07 реальная 2020-12-07 00:00:00
дата остановки набора: ожидаемая 2020-12-21 реальная 2020-12-21
00:00:00
```

Добавим условие чтобы дата остановки набора была не больше 21

```
needed_user_filter_time = []
for elem in (final_ab_new_users.query('first_date < "2020-12-22"')
['user_id']):
            needed_user_filter_time.append(elem)

final_ab_participants_filtered =
final_ab_participants_filtered.query('user_id in
@needed_user_filter_time')
```

дата остановки: 2021-01-04 - НЕ ВЕРНО (нет данных за послдении пять дней, 30 число доступно лишь до 12:00)

```
print('дата полной остановки:', 'ожидаемая 2021-01-04', 'реальная',
     final_ab_events[final_ab_events.user_id.isin(needed_user_filter)
== True]['event_dt'].max())
```

```
дата полной остановки: ожидаемая 2021-01-04 реальная 2020-12-30
06:42:52
```

В случае полной остановки на 5 дней раньше, мы потеряем данные о части действий пользователей набранных после 16 декабря. Их лайфтам будет не полным, это может сделать результат теста неточным

аудитория: в тест должно быть отобрано 15% новых пользователей из региона EU - ВЕРНО

```
needed_user_recommender = []
for elem in
(final_ab_participants['user_id'].loc[(final_ab_participants['ab_test'
] == 'recommender_system_test')]):
   needed_user_recommender.append(elem)

print('доля пользователей из EU', final_ab_new_users.query('region ==
"EU"').query('user_id in @needed_user_recommender')['user_id'].count()
```

```
          / final_ab_new_users.query('region == "EU"').query('user_id
in @needed_user_filter_time')['user_id'].count() * 100)

доля пользователей из EU 15.0
```

ожидаемое количество участников теста: 6000 - Не верно (всего 5231)

```
#проверим данные на дубликаты
final_ab_participants_filtered =
final_ab_participants_filtered.drop_duplicates().reset_index(drop =
True)
final_ab_participants_filtered.query('ab_test ==
"recommender_system_test" ')

               user_id group                  ab_test
0     D1ABA3E2887B6A73     A  recommender_system_test
1     A7A3664BD6242119     A  recommender_system_test
2     DABC14FDDFADD29E     A  recommender_system_test
3     04988C5DF189632E     A  recommender_system_test
4     482F14783456D21B     B  recommender_system_test
...                ...   ...                     ...
5226  A23B0A7FFF375BFF     B  recommender_system_test
5227  7C5C12FA1B5AB710     A  recommender_system_test
5228  91C3969B8A72B908     B  recommender_system_test
5229  E26F13A65CEAC6EA     A  recommender_system_test
5230  80712ED4EA1B52A5     A  recommender_system_test

[5231 rows x 3 columns]

print('всего в тесте:',
len(final_ab_participants_filtered.query('ab_test ==
"recommender_system_test"')))

всего в тесте: 5231
```

убедимся, что время проведения теста не совпадает с маркетинговыми и другими активностями

время проведения совпадает с акциией

Christmas&New Year Promo EU, N.America 2020-12-25 2021-01-03

```
print(ab_project_marketing_events)

                                name                   regions
start_dt  \
0           Christmas&New Year Promo             EU, N.America 2020-
12-25
1       St. Valentine's Day Giveaway  EU, CIS, APAC, N.America 2020-
02-14
```

```
2             St. Patric's Day Promo             EU, N.America 2020-
03-17
3                       Easter Promo  EU, CIS, APAC, N.America 2020-
04-12
4                   4th of July Promo                 N.America 2020-
07-04
5          Black Friday Ads Campaign  EU, CIS, APAC, N.America 2020-
11-26
6              Chinese New Year Promo                      APAC 2020-
01-25
7   Labor day (May 1st) Ads Campaign             EU, CIS, APAC 2020-
05-01
8    International Women's Day Promo             EU, CIS, APAC 2020-
03-08
9    Victory Day CIS (May 9th) Event                      CIS 2020-
05-09
10         CIS New Year Gift Lottery                      CIS 2020-
12-30
11     Dragon Boat Festival Giveaway                      APAC 2020-
06-25
12           Single's Day Gift Promo                      APAC 2020-
11-11
13              Chinese Moon Festival                     APAC 2020-
10-01

    finish_dt
0  2021-01-03
1  2020-02-16
2  2020-03-19
3  2020-04-19
4  2020-07-11
5  2020-12-01
6  2020-02-07
7  2020-05-03
8  2020-03-10
9  2020-05-11
10 2021-01-07
11 2020-07-01
12 2020-11-12
13 2020-10-07
```

проверим равномерность распределения по тестовым группам и правильность их формирования

```
print('всего пользователей по группам:',
final_ab_participants_filtered.query('ab_test ==
"recommender_system_test" ').groupby('group')['user_id'].count())
```

```
всего пользователей по группам: group
A    3195
B    2036
Name: user_id, dtype: int64
```

Неравномерное распределение пользователей между группами может привести к смещению результатов теста. Группа с большим количеством пользователей может иметь большую статистическую мощность, что означает, что даже небольшие различия между группами могут быть обнаружены как статистически значимые. В то же время, группа с меньшим количеством пользователей может иметь меньшую статистическую мощность, что делает обнаружение различий более сложным.

проверим равномерность распределения по регионам. Используя эти данные можно смело игнорировать промо, оно влияет на весь датасет(EU).

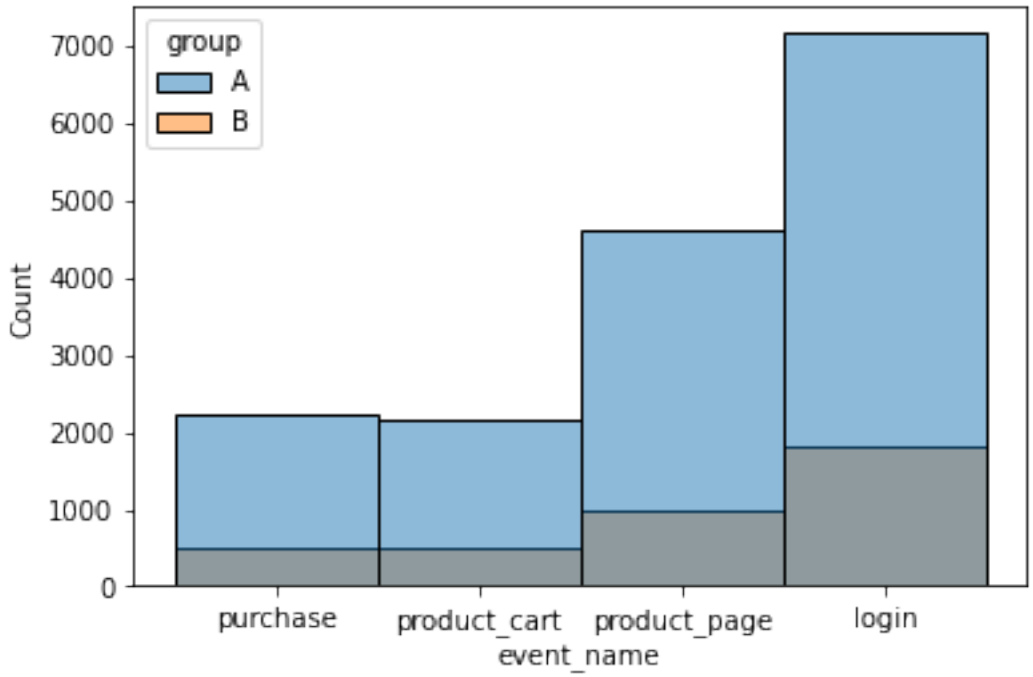
```
needed_user_filter = []
for elem in (
       final_ab_participants_filtered.query('ab_test ==
"recommender_system_test" ')['user_id']):
   needed_user_filter.append(elem)

print(final_ab_new_users[final_ab_new_users.user_id.isin(needed_user_f
ilter) == True].groupby('region')['user_id'].count())

region
EU    5231
Name: user_id, dtype: int64
```

Только пользователи из EU, все верно

проверим равномерность распределения событий по выборкам

```
final_ab_participants_filtered_A =
list(final_ab_participants_filtered.query('group == "A"')['user_id'])
final_ab_participants_filtered_B =
list(final_ab_participants_filtered.query('group == "B"')['user_id'])

df_A = final_ab_events.query('user_id in
@final_ab_participants_filtered_A')
df_B = final_ab_events.query('user_id in
@final_ab_participants_filtered_B')

pd.set_option('mode.chained_assignment', None)
df_A['group'] = 'A'
df_B['group'] = 'B'
df_filtered = pd.concat([df_A, df_B])
```

```
sns.histplot(data=df_filtered, x='event_name', hue='group', bins=50)
plt.show()
```



В проверках наших гипотез мы будем использовать Т-тест чтобы проверить, отличается ли среднее одной переменной от среднего другой переменной.

Чтобы его провести данные должны отвечать следующим критериям:

1)случайная выборка

2)в каждой выборке нормальное распредление и они больше 30 (Благодаря ЦПТ, если размер выборки составляет хотя бы несколько десятков значений, выборочные средние, которые можно получить из одной и той же генеральной совокупности, будут распределены нормально вокруг истинного среднего этой совокупности)

3)они не зависят друг от друга

4)дисперсии должны быть равны(При этом если выборки достаточно велики (30 и больше значений) и равны по размеру между собой, то такой подход оправдан: симуляции, проведённые учёными, показывают, что даже если дисперсии на самом деле не равны, то тест редко ошибается.)

В нашем случае выборки разного размера, на всякий случай укажем equal_var = False

/////////

проверить гипотезу 1: "различия кол-ва событий между группами А и Б."

нулевая гипотеза "кол-во событий клиентов группы А и клиентов группы Б одинаковы"

альтернативная гипотеза "кол-во событий клиентов группы А и клиентов группы Б отличаются"

```
df_A['total_orders'] = df_A.groupby('user_id')
['event_name'].transform('count')
df_B['total_orders'] = df_B.groupby('user_id')
['event_name'].transform('count')
```

Рассмотрим среднее и медиану групп

```
print('среднее // медиана группы A', df_A.groupby('user_id')
['total_orders'].mean().mean(), '//', df_A.groupby('user_id')
['total_orders'].median().median())
print('среднее // медиана группы B', df_B.groupby('user_id')
['total_orders'].mean().mean(), '//', df_B.groupby('user_id')
['total_orders'].median().median())
```

```
среднее // медиана группы A 7.088196577446248 // 6.0
среднее // медиана группы B 5.777099236641221 // 5.0
```

В группе В меньше заказов по обеим метрикам

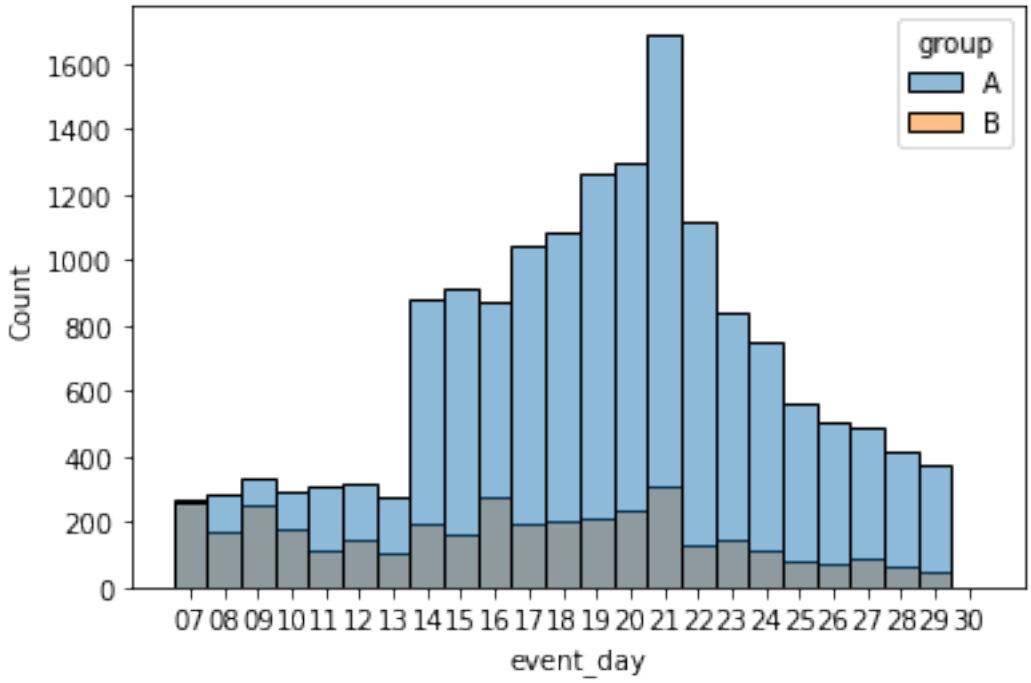
```
alpha = 0.05

results = st.ttest_ind(df_A['total_orders'], df_B['total_orders'],
equal_var = False)

print('p-значение:', results.pvalue)

if results.pvalue < alpha:
    print("Отвергаем нулевую гипотезу")
else:
    print("Не получилось отвергнуть нулевую гипотезу")
```

```
p-значение: 2.460369951110683e-59
Отвергаем нулевую гипотезу
```

В данном случае, значение p-value значительно меньше 0.05, что говорит о том, что нулевая гипотеза отвергается. Таким образом, мы можем сделать вывод о том, что есть статистически значимые различия между количеством заказов клиентов группы А и клиентов группы Б

как число событий в выборках распределено по дням?

```
df_filtered['event_day'] = df_filtered['event_dt'].dt.strftime('%d')
sns.histplot(data=df_filtered, x='event_day', hue='group', bins=50)
plt.show()
```



создаем воронку по группам

```
funnel_a = df_A.groupby('event_name')
['user_id'].nunique().sort_values(ascending=False).to_frame().reset_in
dex()\
        .rename(columns={'user_id': 'total_users'})
funnel_b = df_B.groupby('event_name')
['user_id'].nunique().sort_values(ascending=False).to_frame().reset_in
dex()\
        .rename(columns={'user_id': 'total_users'})
```

Поменяем местами события purchase и product cart в соответствии с логикой пути клиента

```
last_row = funnel_a.iloc[-1].copy()
pre_last_row = funnel_a.iloc[-2].copy()

funnel_a.iloc[-1] = pre_last_row
funnel_a.iloc[-2] = last_row
```

```
last_row = funnel_b.iloc[-1].copy()
pre_last_row = funnel_b.iloc[-2].copy()

funnel_b.iloc[-1] = pre_last_row
funnel_b.iloc[-2] = last_row

fig = go.Figure()

fig.add_trace(go.Funnel(
    name = 'a',
    y = funnel_a['event_name'],
    x = funnel_a['total_users'],
    ))

fig.add_trace(go.Funnel(
    name = 'b',
    y = funnel_b['event_name'],
    x = funnel_b['total_users'],
    ))

fig.show()
```


```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"name":"a","type":"funnel","x":[2279,1476,686,734],"y":
["login","product_page","product_cart","purchase"]},
{"name":"b","type":"funnel","x":[655,367,184,191],"y":
["login","product_page","product_cart","purchase"]}],"layout":
{"template":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
```

```
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter"}],"scatter3d":
[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
```

```
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
```

{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}}}}

Воронки групп практически не различаются

Проведем оценку результатов A/B-тестирования:

проверить гипотезу 1: "различия среднего кол-ва переходов на страницу продукта между группами А и Б."

нулевая гипотеза "среднее кол-во переходов на страницу продукта клиентов группы А и клиентов группы Б одинаковы"

альтернативная гипотеза "среднее кол-во переходов на страницу продукта клиентов группы А и клиентов группы Б отличаются"

проверить гипотезу 2: "различия среднего кол-ва переходов в корзину между группами А и Б."

нулевая гипотеза "среднее кол-во переходов в корзину клиентов группы А и клиентов группы Б одинаковы"

альтернативная гипотеза "среднее кол-во переходов в корзину клиентов группы А и клиентов группы Б отличаются"

проверить гипотезу 3: "различия среднего кол-ва покупок между группами А и Б."

нулевая гипотеза "среднее кол-во покупок клиентов группы А и клиентов группы Б одинаковы"

альтернативная гипотеза "среднее кол-во покупок клиентов группы А и клиентов группы Б отличаются"

//////////////

создадим сводную таблицу с воронкой по группам

```
all_funnels = df_filtered.pivot_table(index='event_name',
columns='group',values='user_id',aggfunc='nunique')\
                        .sort_values('A',ascending=False).T
all_funnels
```

```
event_name  login  product_page  purchase  product_cart
group
```

```
A            2279          1476       734           686
B             655           367       191           184
```

Воспользуемся Z-критерием для проверки результатов теста

```
def z_test(alpha, exp1, exp2, total1, total2):
    p1 = exp1/total1
    p2 = exp2/total2
    p_combined = (exp1 + exp2) / (total1 + total2)
    difference = p1 - p2
    z_value = difference / mth.sqrt(p_combined * (1 - p_combined) * (1
/ total1 + 1 / total2))
    distr = st.norm(0, 1)
    p_value = (1 - distr.cdf(abs(z_value))) * 2
    print('p-значение: ', p_value)
    adjusted_alpha = alpha / 3  # Поправка Бонферонни для трех
проверок

    if (p_value < adjusted_alpha):
        print("Отвергаем нулевую гипотезу")
    else:
        print("Не получилось отвергнуть нулевую гипотезу, доли не
различаются")

z_test(0.05, all_funnels['product_page'][0],
all_funnels['product_page'][1], df_A['user_id'].nunique(),
df_B['user_id'].nunique())
```

```
p-значение:  4.570197193798364e-05
Отвергаем нулевую гипотезу
```

```
z_test(0.05, all_funnels['product_cart'][0],
all_funnels['product_cart'][1], df_A['user_id'].nunique(),
df_B['user_id'].nunique())
```

```
p-значение:  0.3210354197858867
Не получилось отвергнуть нулевую гипотезу, доли не различаются
```

```
z_test(0.05, all_funnels['purchase'][0], all_funnels['purchase'][1],
df_A['user_id'].nunique(), df_B['user_id'].nunique())
```

```
p-значение:  0.13910551275309246
Не получилось отвергнуть нулевую гипотезу, доли не различаются
```

Статистически отличаются переходы на страницу покупок. Доли перехода в корзину и на страницу продукта не различаются.

Выводы:

В тестирование не попала информация о действиях пользователей набранных после 30.12(5 дней).

В группах разное кол-во пользователей.

В группах по разному распределено кол-во заказов.

Большинство остальных пунктов ТЗ соблюдено, или имеют небольшие несответсвия не влияющие на обший результат, например, проведение теста с акцией в одно время(практчиески все пользвоатели тестирования под влиянием).

Неравномерное распределение пользователей между группами A и B влияет на результаты теста. Большая группа (A) имеет большую статистическую мощность, что может привести к обнаружению даже незначительных различий как статистически значимых. Группа с меньшим количеством пользователей (B) имеет меньшую статистическую мощность, что затрудняет обнаружение различий.

Следует признать тестирование некорректным и провести новое.