# Описание проекта

В вашем распоряжении есть датасет с действиями пользователей, техническое задание и несколько вспомогательных датасетов. Оценим корректность проведения теста и проанализируем его результаты.

Чтобы оценить корректность проведения теста:

* удостоверимся, что нет пересечений с конкурирующим тестом и нет пользователей, участвующих в двух группах теста одновременно;
* проверьим равномерность распределения пользователей по тестовым группам и правильность их формирования.

# Описание данных

/datasets/ab\_project\_marketing\_events.csv — календарь маркетинговых событий на 2020 год:

* name — название маркетингового события;
* regions — регионы, в которых будет проводиться рекламная кампания;
* start\_dt — дата начала кампании;
* finish\_dt — дата завершения кампании.

/datasets/final\_ab\_new\_users.csv — все пользователи, зарегистрировавшиеся в интернет-магазине в период с 7 по 21 декабря 2020 года:

* user\_id — идентификатор пользователя;
* first\_date — дата регистрации;
* region — регион пользователя;
* device — устройство, с которого происходила регистрация.

/datasets/final\_ab\_events.csv — все события новых пользователей в период с 7 декабря 2020 по 4 января 2021 года:

* user\_id — идентификатор пользователя;
* event\_dt — дата и время события;
* event\_name — тип события;
* details — дополнительные данные о событии. Например, для покупок, purchase, в этом поле хранится стоимость покупки в долларах.

/datasets/final\_ab\_participants.csv — таблица участников тестов.

* user\_id — идентификатор пользователя;
* ab\_test — название теста;
* group — группа пользователя.

# Техническое задание

* название теста: recommender\_system\_test;
* группы: А — контрольная, B — новая;
* дата запуска: 2020-12-07;
* дата остановки набора новых пользователей: 2020-12-21;
* дата остановки: 2021-01-04;
* ожидаемое количество участников теста: 15% новых пользователей из региона EU;
* назначение теста: тестирование изменений, связанных с внедрением улучшенной рекомендательной системы;
* ожидаемый эффект: за 14 дней с момента регистрации в системе пользователи покажут улучшение каждой метрики не менее, чем на 5 процентных пунктов: конверсии в просмотр карточек товаров — событие product\_page, просмотры корзины — product\_cart, покупки — purchase.

# План анализа

* ознакомление с данными и первичная фильтрация;
* оценка корректонсти проведения теста, проверка на соответствие данных ТЗ;
* исследовательский анализ данных;
* оценка результатов А/В-теста;
* выводы.

## 1  Ознакомление с данными и первичная фильтрация.

Ввод [1]:

*#импорт библиотек*

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

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

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

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

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

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

**import** datetime **as** dt

**import** plotly.graph\_objs **as** go

**from** statsmodels.stats.proportion **import** proportions\_ztest

**import** warnings

warnings.simplefilter('ignore')

Ввод [2]:

*#чтение файлов, сохранение в датафсеты*

marketing\_events **=** pd.read\_csv('/datasets/ab\_project\_marketing\_events.csv')

new\_users **=** pd.read\_csv('/datasets/final\_ab\_new\_users.csv')

events **=** pd.read\_csv('/datasets/final\_ab\_events.csv')

participants **=** pd.read\_csv('/datasets/final\_ab\_participants.csv')

​

Ввод [3]:

*#ознакомимся с общей информацией*

marketing\_events.info()

new\_users.info()

events.info()

participants.info()

<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

<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

<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

<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

Ввод [4]:

*#все столбцы с датой имеют некорректный тип данных, исправим это*

marketing\_events['start\_dt'] **=** pd.to\_datetime(marketing\_events['start\_dt'])

marketing\_events['finish\_dt'] **=** pd.to\_datetime(marketing\_events['finish\_dt'])

new\_users['first\_date'] **=** pd.to\_datetime(new\_users['first\_date'])

events['event\_dt'] **=** pd.to\_datetime(events['event\_dt'])

Ввод [5]:

*#проверим изменения*

marketing\_events.info()

<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 datetime64[ns]

3 finish\_dt 14 non-null datetime64[ns]

dtypes: datetime64[ns](2), object(2)

memory usage: 576.0+ bytes

Ввод [6]:

*#проверим данные на пропуски*

null\_marketing\_events **=** marketing\_events.isnull().sum()

null\_new\_users **=** new\_users.isnull().sum()

null\_events **=** events.isnull().sum()

null\_participants **=** participants.isnull().sum()

​

null\_marketing\_events, null\_new\_users, null\_events, null\_participants

Out[6]:

(name 0

regions 0

start\_dt 0

finish\_dt 0

dtype: int64,

user\_id 0

first\_date 0

region 0

device 0

dtype: int64,

user\_id 0

event\_dt 0

event\_name 0

details 377577

dtype: int64,

user\_id 0

group 0

ab\_test 0

dtype: int64)

В датафрейме events есть пропуске в столбце details, который содержит в себе дополнительные данные, например сумму покупки.

Ввод [7]:

*#проверим данные на дубликаты*

​

print(marketing\_events.duplicated().sum(), new\_users.duplicated().sum(), events.duplicated().sum(), participants.duplicated().sum())

0 0 0 0

Ввод [8]:

*#перезапишем датафреймы с учетом ТЗ*

marketing\_events **=** marketing\_events.loc[marketing\_events['regions'].str.contains('EU')]

marketing\_events

Out[8]:

|  | **name** | **regions** | **start\_dt** | **finish\_dt** |
| --- | --- | --- | --- | --- |
| **0** | Christmas&New Year Promo | EU, N.America | 2020-12-25 | 2021-01-03 |
| **1** | St. Valentine's Day Giveaway | EU, CIS, APAC, N.America | 2020-02-14 | 2020-02-16 |
| **2** | St. Patric's Day Promo | EU, N.America | 2020-03-17 | 2020-03-19 |
| **3** | Easter Promo | EU, CIS, APAC, N.America | 2020-04-12 | 2020-04-19 |
| **5** | Black Friday Ads Campaign | EU, CIS, APAC, N.America | 2020-11-26 | 2020-12-01 |
| **7** | Labor day (May 1st) Ads Campaign | EU, CIS, APAC | 2020-05-01 | 2020-05-03 |
| **8** | International Women's Day Promo | EU, CIS, APAC | 2020-03-08 | 2020-03-10 |

Ввод [9]:

new\_users **=** new\_users.loc[(new\_users['region']**==**'EU')**&**(new\_users['first\_date']**>=**'2020-12-07')**&**(new\_users['first\_date']**<=**'2020-12-21')]

new\_users.head()

Out[9]:

|  | **user\_id** | **first\_date** | **region** | **device** |
| --- | --- | --- | --- | --- |
| **0** | D72A72121175D8BE | 2020-12-07 | EU | PC |
| **2** | 2E1BF1D4C37EA01F | 2020-12-07 | EU | PC |
| **3** | 50734A22C0C63768 | 2020-12-07 | EU | iPhone |
| **7** | 8942E64218C9A1ED | 2020-12-07 | EU | PC |
| **9** | FFCEA1179C253104 | 2020-12-07 | EU | Android |

Ввод [10]:

events **=** events.loc[(events['event\_dt'] **>=** '2020-12-07') **&** (events['event\_dt'] **<=** '2021-01-04')]

events.head()

Out[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 |

Ввод [11]:

*#исследуем пропуски в разрезе уникальных event\_name*

events.groupby('event\_name')['details'].apply(**lambda** x: x.isnull().sum())

Out[11]:

event\_name

login 189552

product\_cart 62462

product\_page 125563

purchase 0

Name: details, dtype: int64

Как и заявлено выше, у нас есть 4 датафрейма marketing\_events (14 строк), new\_users(61733 строк), events (440317 строк) и participants (18268 строк) с некорректным типом данных в дате - скорректировано; и с пропусками в датафрейме events в столбце details, который содержит в себе дополнительные данные, например сумму покупки (все остальные данные там не заполнялись судя по всему); дубликаты не найдены.

## 2  Oценка корректнoсти проведения теста, проверка на соответствие данных ТЗ.

Ввод [12]:

*#выделим пользователей, учавствовавших в тесте*

recommender\_participants **=** participants.loc[participants['ab\_test'] **==** 'recommender\_system\_test']

recommender\_participants.head()

Out[12]:

|  | **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 |

Мы обладаем пользователями из нашего теста recommender\_system\_test, разделенными на группы; по датам и региону из ТЗ датафреймы были отфильтрованы ранее.

Ввод [13]:

*#составляет ли общее количество пользователей из целевого региона 15% от общего числа пользователей из целевого региона,*

*#зарегистрированных в период набора пользователей в тест?*

​

*#объеденим датасеты*

merged\_df **=** recommender\_participants.merge(new\_users, on**=**'user\_id')

*#проверим долю*

share **=** merged\_df['user\_id'].count()**/**new\_users.loc[new\_users['region']**==**'EU'].count()[0]**\***100

share

Out[13]:

15.0

Ровно 15% составляют новые пльзователи из целевого региона.

Ввод [14]:

*#проверка на наличие людей, участвующих в нескольких тестах*

multi\_test\_participants **=** participants.groupby('user\_id')['ab\_test'].nunique() **>** 1

print("Количество людей, участвующих в нескольких тестах:", multi\_test\_participants.sum())

Количество людей, участвующих в нескольких тестах: 1602

Ввод [15]:

*#подсчет доли таких людей от общего количества клиентов в тесте recommender\_sistem\_test*

multi\_test\_share **=** recommender\_participants['user\_id'].count()**/**multi\_test\_participants.sum()

multi\_test\_share.round(2)

Out[15]:

4.18

Доля таких клиентов составляет около 25% - и это очень много для того, чтобы просто взять и удалить их.

Ввод [16]:

*#проверим есть ли у нас пользователи, которые попали в обе группы теста*

duplicated\_users **=** recommender\_participants.groupby('user\_id').filter(**lambda** x: len(x['group'].unique()) **==** 2)['user\_id'].unique()

​

**if** len(duplicated\_users) **>** 0:

print("Найдены пользователи с дублирующимися значениями group:", duplicated\_users)

**else**:

print("Нет пользователей с дублирующимися значениями group.")

Нет пользователей с дублирующимися значениями group.

Ввод [17]:

*# подсчет количества пользователей в каждой группе*

group\_sizes **=** recommender\_participants.groupby('group').size().reset\_index(name**=**'count')

​

*# подсчет общего количества пользователей*

total\_users **=** len(recommender\_participants['user\_id'].unique())

​

*# добавление столбца с долями пользователей в каждой группе*

group\_sizes['share'] **=** group\_sizes['count'] **/** total\_users

​

*# вывод таблицы*

group\_sizes

Out[17]:

|  | **group** | **count** | **share** |
| --- | --- | --- | --- |
| **0** | A | 3824 | 0.570661 |
| **1** | B | 2877 | 0.429339 |

Ввод [18]:

*#составим объединенный датафрейм для дальнейшего анализа*

merged\_df\_2 **=** merged\_df.merge(events, on**=**'user\_id', how**=**'left')

merged\_df\_2.head()

Out[18]:

|  | **user\_id** | **group** | **ab\_test** | **first\_date** | **region** | **device** | **event\_dt** | **event\_name** | **details** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-07 14:43:27 | purchase | 99.99 |
| **1** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-25 00:04:56 | purchase | 4.99 |
| **2** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-07 14:43:29 | product\_cart | NaN |
| **3** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-25 00:04:57 | product\_cart | NaN |
| **4** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-07 14:43:27 | product\_page | NaN |

Ввод [19]:

*#добавим новый столбец с возрастом каждого события*

merged\_df\_2['event\_age'] **=** merged\_df\_2['event\_dt'] **-** merged\_df\_2['first\_date']

merged\_df\_2.head()

Out[19]:

|  | **user\_id** | **group** | **ab\_test** | **first\_date** | **region** | **device** | **event\_dt** | **event\_name** | **details** | **event\_age** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-07 14:43:27 | purchase | 99.99 | 0 days 14:43:27 |
| **1** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-25 00:04:56 | purchase | 4.99 | 18 days 00:04:56 |
| **2** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-07 14:43:29 | product\_cart | NaN | 0 days 14:43:29 |
| **3** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-25 00:04:57 | product\_cart | NaN | 18 days 00:04:57 |
| **4** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-07 14:43:27 | product\_page | NaN | 0 days 14:43:27 |

Ввод [20]:

*#задаем горизонт*

horizon **=** pd.Timedelta(days**=**14)

*#фильтруем датафрейм по событиям, чей возраст соответсвует горизонту или менее*

merged\_df\_2 **=** merged\_df\_2[merged\_df\_2['event\_age'] **<=** horizon].copy()

*#оставляем в столбце только значение дней*

merged\_df\_2['event\_age'] **=** merged\_df\_2['event\_age'].dt.days

merged\_df\_2.head()

Out[20]:

|  | **user\_id** | **group** | **ab\_test** | **first\_date** | **region** | **device** | **event\_dt** | **event\_name** | **details** | **event\_age** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-07 14:43:27 | purchase | 99.99 | 0 |
| **2** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-07 14:43:29 | product\_cart | NaN | 0 |
| **4** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-07 14:43:27 | product\_page | NaN | 0 |
| **6** | D1ABA3E2887B6A73 | A | recommender\_system\_test | 2020-12-07 | EU | PC | 2020-12-07 14:43:27 | login | NaN | 0 |
| **8** | A7A3664BD6242119 | A | recommender\_system\_test | 2020-12-20 | EU | iPhone | 2020-12-20 15:46:06 | product\_page | NaN | 0 |

Ввод [21]:

print(merged\_df\_2['event\_name'].unique())

['purchase' 'product\_cart' 'product\_page' 'login']

Данные содержат инфо о пользователях, принявших участие в двух тестах, один из которых наш - recommender\_system\_test - результаты которого мы проанализируем ниже. Мы отфильтровали по нему данные и убедились, что пользователи разделены на группы А и В и присутствуют в них в достаточном количестве, хоть их распределение и сложно назвать равномерным. Пользователи, которые попали бы в обе группы не найдены, распределены корректно. Среди уникальных событий в логе: оплата, продуктовая корзина, карточка товара, регистрация.

## 3  Исследовательский анализ данных.

Ввод [22]:

*# Создаем два датафрейма для каждой группы*

group\_a **=** merged\_df\_2[merged\_df\_2['group'] **==** 'A']

group\_b **=** merged\_df\_2[merged\_df\_2['group'] **==** 'B']

​

*# Считаем количество событий на 1 пользователя для каждой группы*

events\_per\_user\_a **=** group\_a.groupby('user\_id')['event\_name'].count()

events\_per\_user\_b **=** group\_b.groupby('user\_id')['event\_name'].count()

​

*# Строим гистограмму для группы A*

plt.hist(events\_per\_user\_a, bins**=**20, alpha**=**0.5, label**=**'Группа A')

*# Строим гистограмму для группы B*

plt.hist(events\_per\_user\_b, bins**=**20, alpha**=**0.5, label**=**'Группа B')

​

plt.legend(loc**=**'upper right')

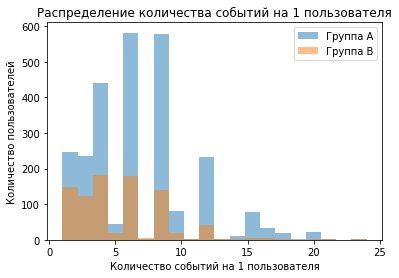
plt.xlabel('Количество событий на 1 пользователя')

plt.ylabel('Количество пользователей')

plt.title('Распределение количества событий на 1 пользователя')

​

plt.show()



Ввод [23]:

*#создаем два датафрейма для каждой группы*

group\_a **=** merged\_df\_2[merged\_df\_2['group'] **==** 'A']

group\_b **=** merged\_df\_2[merged\_df\_2['group'] **==** 'B']

*#выводим среднее количество событий на пользователя по каждой из групп*

print(group\_a.groupby('user\_id')['event\_name'].count().mean())

print(group\_b.groupby('user\_id')['event\_name'].count().mean())

6.849078341013825

5.456100342075256

В группе А чаще всего 1 пользователь совершает 6 или 8 событий, переходя от одного к другому; в группе В - 4или 6 событий; в обеих группах меньше всего пользователей, которые предпочитают совершить количество шагов свыше 15, но есть и такие кто способен и на 25 шагов за раз! Среднее же количество событий на пользователя по каждой из групп: для А - 6,8, а для В - 5,4 события.

Ввод [24]:

merged\_df\_2['date'] **=** merged\_df\_2['event\_dt'].dt.date

plt.figure(figsize**=**(10, 5))

sns.histplot(x**=**'event\_dt', hue**=**'group', data**=**merged\_df\_2, multiple**=**"stack", palette**=**'bright')

​

plt.ylabel('Количество событий')

plt.xlabel('Даты')

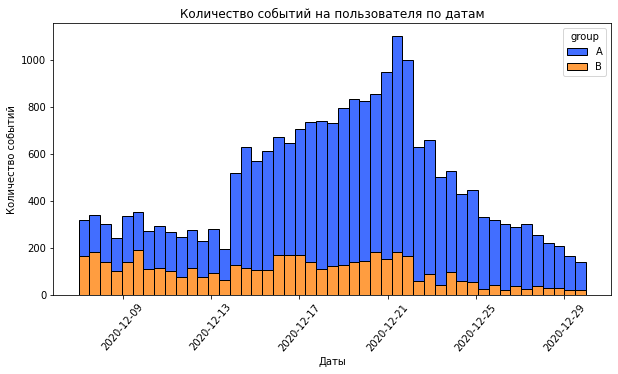
plt.xticks(rotation**=**50)

plt.title('Количество событий на пользователя по датам')

​

Out[24]:

Text(0.5, 1.0, 'Количество событий на пользователя по датам')



После 14/12/2020 количество событий на 1 пользователя в день в группе А начинает расти и достигает своего пика к 22/12/2020 - к концу анализируемого периода, затем падает. По группе В наибольшее количество событий приходится на начало анаизируемого периода, возрастает незначительно к середине периода и начинает стабильно падать после - количество событий распределено более равномерно, чем в группе А.

Ввод [25]:

*#проверим проходили ли маркетинговые активности одновременно с нашим тестом*

start **=** dt.date(2020, 12, 7)

finish **=** dt.date(2021, 1, 4)

*#маркетинговые активности от начала до конца теста*

marketing\_events.query('not(start\_dt > @finish or finish\_dt < @start)')

Out[25]:

|  | **name** | **regions** | **start\_dt** | **finish\_dt** |
| --- | --- | --- | --- | --- |
| **0** | Christmas&New Year Promo | EU, N.America | 2020-12-25 | 2021-01-03 |

**Комментарий cтудента**

Ввод [26]:

*# Отфильтровываем датафрейм по группе A и построим простую продуктовую воронку с логической последовательностью событий*

group\_a **=** merged\_df\_2[merged\_df\_2['group'] **==** 'A']

​

*# Считаем количество уникальных пользователей для каждого события*

event\_1\_count **=** group\_a[group\_a['event\_name'] **==** 'login']['user\_id'].nunique()

event\_2\_count **=** group\_a[group\_a['event\_name'] **==** 'product\_page']['user\_id'].nunique()

event\_3\_count **=** group\_a[group\_a['event\_name'] **==** 'product\_cart']['user\_id'].nunique()

event\_4\_count **=** group\_a[group\_a['event\_name'] **==** 'purchase']['user\_id'].nunique()

​

*# Создаем список с названиями событий*

events **=** ['login', 'product\_page', 'product\_cart', 'purchase']

​

*# Создаем список с количеством пользователей для каждого события*

users\_counts **=** [event\_1\_count, event\_2\_count, event\_3\_count, event\_4\_count]

​

*# Создаем датафрейм с данными для воронки*

funnel\_data **=** pd.DataFrame({'events': events, 'users\_counts': users\_counts})

​

*# Отображаем воронку*

fig **=** go.Figure(go.Funnel(

y**=**funnel\_data['events'],

x**=**funnel\_data['users\_counts'],

textposition**=**'inside',

marker**=**dict(

color**=**funnel\_data['users\_counts'],

colorscale**=**'RdBu',

reversescale**=True**,

showscale**=True**

)

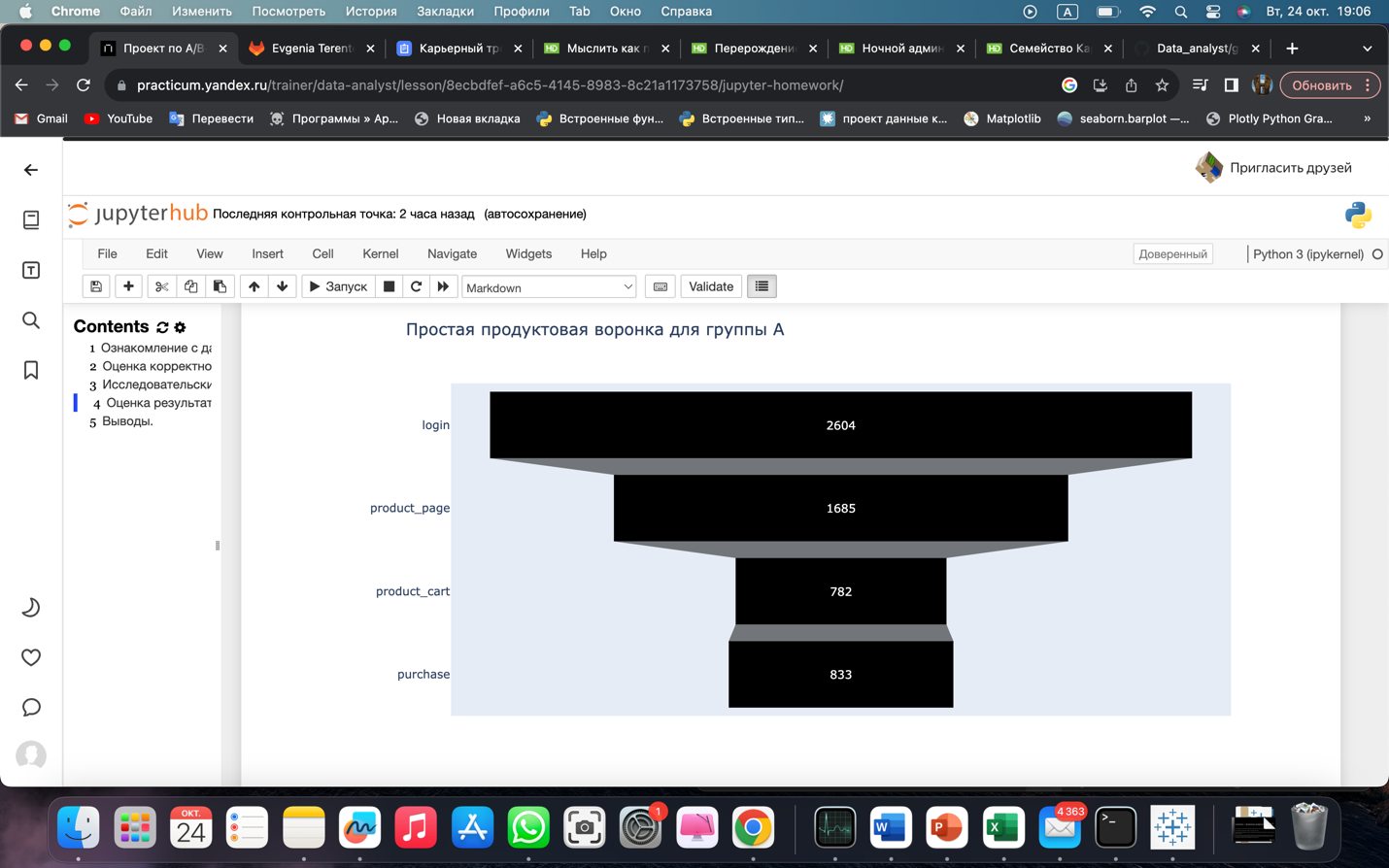
))

fig.update\_layout(title**=**'Простая продуктовая воронка для группы А')

fig.show()

26041685782833purchaseproduct\_cartproduct\_pagelogin

Простая продуктовая воронка для группы А



Ввод [27]:

*# Отфильтровываем датафрейм по группе В и построим простую продуктовую воронку с логической последовательностью событий*

group\_a **=** merged\_df\_2[merged\_df\_2['group'] **==** 'B']

​

*# Считаем количество уникальных пользователей для каждого события*

event\_1\_count **=** group\_a[group\_a['event\_name'] **==** 'login']['user\_id'].nunique()

event\_2\_count **=** group\_a[group\_a['event\_name'] **==** 'product\_page']['user\_id'].nunique()

event\_3\_count **=** group\_a[group\_a['event\_name'] **==** 'product\_cart']['user\_id'].nunique()

event\_4\_count **=** group\_a[group\_a['event\_name'] **==** 'purchase']['user\_id'].nunique()

​

*# Создаем список с названиями событий*

events **=** ['login', 'product\_page', 'product\_cart', 'purchase']

​

*# Создаем список с количеством пользователей для каждого события*

users\_counts **=** [event\_1\_count, event\_2\_count, event\_3\_count, event\_4\_count]

​

*# Создаем датафрейм с данными для воронки*

funnel\_data **=** pd.DataFrame({'events': events, 'users\_counts': users\_counts})

​

*# Отображаем воронку*

fig **=** go.Figure(go.Funnel(

y**=**funnel\_data['events'],

x**=**funnel\_data['users\_counts'],

textposition**=**'inside',

marker**=**dict(

color**=**funnel\_data['users\_counts'],

colorscale**=**'RdBu',

reversescale**=True**,

showscale**=True**

)

))

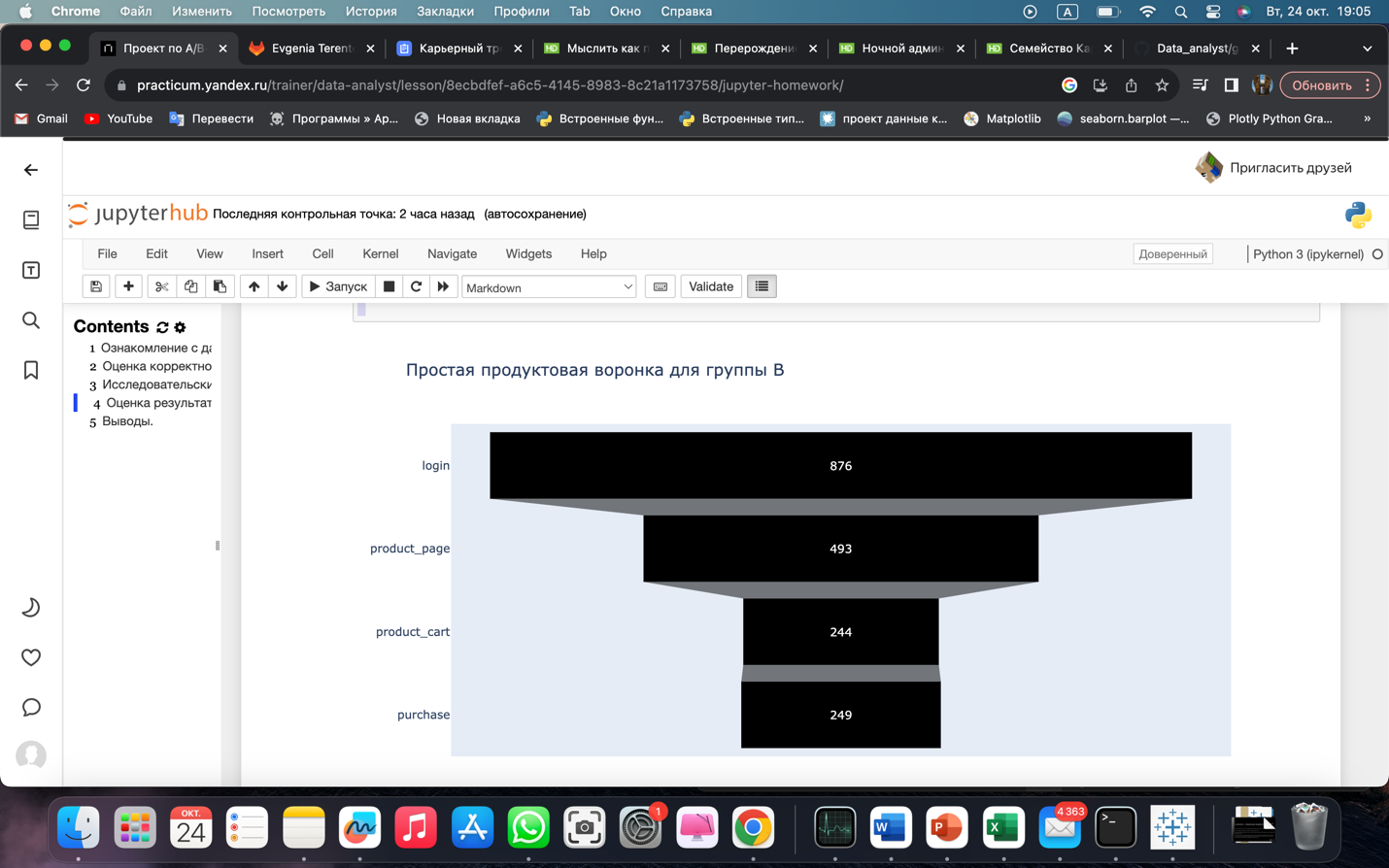
fig.update\_layout(title**=**'Простая продуктовая воронка для группы В')

fig.show()

​

876493244249purchaseproduct\_cartproduct\_pagelogin

Простая продуктовая воронка для группы В



Пропорционально обе воронки выглядят одинаково, но! Конверсия по группам значительно лучше в группе А, ожидаемый эффект по группе В не наблюдается. Так же, сновное количество событий приходится на первые дни, а после 4-5 дней оно идет на спад: таким образом 14 дней достаточно для теста. Судя по графикам, пользовательская активность группы В ниже актвности, группы А.

## 4  Oценка результатов А/В-теста.

Проверим, являются ли различия между группами статистически значимыми.

Н0: Доли не различаются статистически значимо.

Н1: Разница между долями статистически значима.

Ввод [28]:

alpha **=** .05**/**3 *# критический уровень статистической значимости с учетом поправки на множественное сравнение*

​

successes **=** np.array([1685, 493])

trials **=** np.array([2604, 876])

​

*# пропорция успехов в первой группе:*

p1 **=** successes[0]**/**trials[0]

​

*# пропорция успехов во второй группе:*

p2 **=** successes[1]**/**trials[1]

​

*# пропорция успехов в комбинированном датасете:*

p\_combined **=** (successes[0] **+** successes[1]) **/** (trials[0] **+** trials[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**/**trials[0] **+** 1**/**trials[1]))

​

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

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

​

p\_value **=** (1 **-** distr.cdf(abs(z\_value))) **\*** 2

​

print('p-значение: ', p\_value)

​

**if** p\_value **<** alpha:

print('Отвергаем нулевую гипотезу: между долями есть значимая разница')

**else**:

print(

'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными'

)

p-значение: 8.195976000324734e-06

Отвергаем нулевую гипотезу: между долями есть значимая разница

Ввод [29]:

alpha **=** .05**/**3 *# критический уровень статистической значимости с поправкой Бонферрони*

​

successes **=** np.array([782, 244])

trials **=** np.array([1685, 493])

​

*# пропорция успехов в первой группе:*

p1 **=** successes[0]**/**trials[0]

​

*# пропорция успехов во второй группе:*

p2 **=** successes[1]**/**trials[1]

​

*# пропорция успехов в комбинированном датасете:*

p\_combined **=** (successes[0] **+** successes[1]) **/** (trials[0] **+** trials[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**/**trials[0] **+** 1**/**trials[1]))

​

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

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

​

p\_value **=** (1 **-** distr.cdf(abs(z\_value))) **\*** 2

​

print('p-значение: ', p\_value)

​

**if** p\_value **<** alpha:

print('Отвергаем нулевую гипотезу: между долями есть значимая разница')

**else**:

print(

'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными'

)

p-значение: 0.2276722007959906

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Ввод [30]:

alpha **=** .05**/**3 *# критический уровень статистической значимости с поправкой Бонферрони*

​

successes **=** np.array([833, 249])

trials **=** np.array([1685, 493])

​

*# пропорция успехов в первой группе:*

p1 **=** successes[0]**/**trials[0]

​

*# пропорция успехов во второй группе:*

p2 **=** successes[1]**/**trials[1]

​

*# пропорция успехов в комбинированном датасете:*

p\_combined **=** (successes[0] **+** successes[1]) **/** (trials[0] **+** trials[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**/**trials[0] **+** 1**/**trials[1]))

​

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

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

​

p\_value **=** (1 **-** distr.cdf(abs(z\_value))) **\*** 2

​

print('p-значение: ', p\_value)

​

**if** p\_value **<** alpha:

print('Отвергаем нулевую гипотезу: между долями есть значимая разница')

**else**:

print(

'Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными'

)

p-значение: 0.6757323469354453

Не получилось отвергнуть нулевую гипотезу, нет оснований считать доли разными

Так как у нас в тесте 3 сравнения то мы использовали поправку Бонферрони для множественных сравнений и применяли ее к alpha. Разница достигла статистической значимости только в одном тесте при сравнении долей просмотревших карточки товара, для остальных - не достигнута. Не отвергаем нулевую гипотезу, группы были сравниваемы по одной метрике.

## 5  Выводы.

В ходе работы над проектом были выполнены такие этапы как: выгрузка и ознакомление с данными; проверка на пропуски и дубликаты; преобразование типов данных; данные проверены на соответствие ТЗ; проведен исследовательский анализ данных; анализ результатов А/В-теста; проверена статистическая разница долей z-критерием.

Были предоставлены 4 датафрейма marketing\_events (14 строк), new\_users(61733 строк), events (440317 строк) и participants (18268 строк) с некорректным типом данных в дате - скорректировано; и с пропусками в датафрейме events в столбце details, который содержит в себе дополнительные данные, например сумму покупки; дубликаты не найдены.

Данные содержат инфо о пользователях, принявших участие в двух тестах, один из которых наш - recommender\_system\_test - результаты которого мы проанализировали. Мы отфильтровали по нему данные и убедились, что пользователи разделены на группы А и В и присутствуют в них в достаточном количестве. Пользователи, которые попали бы в обе группы не найдены. Среди уникальных событий в логе: оплата, продуктовая корзина, карточка товара, регистрация.

Среднее количество событий на человека в группе А - 6,8, в группе В - 5,4. Анализируемый период - 14 дней. Судя по графикам, пользовательская активность группы В ниже активности, группы А. Пропорционально обе воронки событий выглядят одинаково, но конверсия по группам значительно лучше в группе А, ожидаемый эффект по группе В не наблюдается. Так же, сновное количество событий приходится на первые дни, а после 4-5 дней оно идет на спад: таким образом 14 дней достаточно для теста.

Так как у нас в тесте 3 сравнения то мы использовали поправку Бонферрони для множественных сравнений и применяли ее к уровню стат значимости alpha. Разница достигла статистической значимости только в одном тесте при сравнении долей просмотревших карточки товара, для остальных - не достигнута: мы не отвергаем нулевую гипотезу.

По проведенному тесту можно сделать вывод, что ожидаемый эффект в изменении конверсии не был достигрут. Считаю корректность проведенного теста достаточной.