# Анализ гипотез для увеличения выручки интернет-магазина.

Привет! Я — аналитик крупного интернет-магазина. Вместе с отделом маркетинга мы подготовили список гипотез для увеличения выручки. Приоритизируем гипотезы, запустим A/B-тест и проанализируем результаты.

# Описание данных

Данные для первой части

Файл /datasets/hypothesis.csv.

* Hypothesis — краткое описание гипотезы;
* Reach — охват пользователей по 10-балльной шкале;
* Impact — влияние на пользователей по 10-балльной шкале;
* Confidence — уверенность в гипотезе по 10-балльной шкале;
* Efforts — затраты ресурсов на проверку гипотезы по 10-балльной шкале. Чем больше значение Efforts, тем дороже проверка гипотезы.

Данные для второй части

Файл /datasets/orders.csv.

* transactionId — идентификатор заказа;
* visitorId — идентификатор пользователя, совершившего заказ;
* date — дата, когда был совершён заказ;
* revenue — выручка заказа;
* group — группа A/B-теста, в которую попал заказ.

Файл /datasets/visitors.csv.

* date — дата;
* group — группа A/B-теста;
* visitors — количество пользователей в указанную дату в указанной группе A/B-теста

# План анализа:

* Часть 1. Приоритизация гипотез. В файле /datasets/hypothesis.csv 9 гипотез по увеличению выручки интернет-магазина с указанными параметрами Reach, Impact, Confidence, Effort - применим к ним фреймворки ICE и RICE.
* Часть 2. Анализ A/B-теста.

Ввод [1]:

*#подключаем все необходимые библиотеки*

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

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

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

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

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

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

**from** pandas.plotting **import** register\_matplotlib\_converters

**import** warnings

# Часть 1. Приоритизация гипотез

Ввод [2]:

*#выгрузка и ознакомление с данными датасета с гипотезами*

hyps **=** pd.read\_csv('/datasets/hypothesis.csv', sep**=**',')

pd.set\_option('max\_colwidth', 150)

​

*#приведем столбцы к нижнему регистру*

hyps.columns **=** hyps.columns.str.lower()

hyps

Out[2]:

|  | **hypothesis** | **reach** | **impact** | **confidence** | **efforts** |
| --- | --- | --- | --- | --- | --- |
| **0** | Добавить два новых канала привлечения трафика, что позволит привлекать на 30% больше пользователей | 3 | 10 | 8 | 6 |
| **1** | Запустить собственную службу доставки, что сократит срок доставки заказов | 2 | 5 | 4 | 10 |
| **2** | Добавить блоки рекомендаций товаров на сайт интернет магазина, чтобы повысить конверсию и средний чек заказа | 8 | 3 | 7 | 3 |
| **3** | Изменить структура категорий, что увеличит конверсию, т.к. пользователи быстрее найдут нужный товар | 8 | 3 | 3 | 8 |
| **4** | Изменить цвет фона главной страницы, чтобы увеличить вовлеченность пользователей | 3 | 1 | 1 | 1 |
| **5** | Добавить страницу отзывов клиентов о магазине, что позволит увеличить количество заказов | 3 | 2 | 2 | 3 |
| **6** | Показать на главной странице баннеры с актуальными акциями и распродажами, чтобы увеличить конверсию | 5 | 3 | 8 | 3 |
| **7** | Добавить форму подписки на все основные страницы, чтобы собрать базу клиентов для email-рассылок | 10 | 7 | 8 | 5 |
| **8** | Запустить акцию, дающую скидку на товар в день рождения | 1 | 9 | 9 | 5 |

Вывод: мы видим 9 гипотез доступных для приоритизации с заявленными выше параметрами Reach (охват пользователей), Impact (влияние на пользователей), Confidence (уверенность в гипотезе) и Efforts (затраты ресурсов на проверку гипотезы).

Посчитаем ICE для гипотез. Добавим полученные значения ICE к существующей таблице. Новый столбец назовем ICE. Выведем столбцы 'hypothesis' и 'ICE', отсортированные по столбцу 'ICE' по убыванию приоритета.

Ввод [3]:

hyps['ICE'] **=** (hyps['impact']**\***hyps['confidence'])**/**hyps['efforts']

iced\_hyps **=** hyps[['hypothesis', 'ICE']].sort\_values('ICE', ascending**=False**)

iced\_hyps

Out[3]:

|  | **hypothesis** | **ICE** |
| --- | --- | --- |
| **8** | Запустить акцию, дающую скидку на товар в день рождения | 16.200000 |
| **0** | Добавить два новых канала привлечения трафика, что позволит привлекать на 30% больше пользователей | 13.333333 |
| **7** | Добавить форму подписки на все основные страницы, чтобы собрать базу клиентов для email-рассылок | 11.200000 |
| **6** | Показать на главной странице баннеры с актуальными акциями и распродажами, чтобы увеличить конверсию | 8.000000 |
| **2** | Добавить блоки рекомендаций товаров на сайт интернет магазина, чтобы повысить конверсию и средний чек заказа | 7.000000 |
| **1** | Запустить собственную службу доставки, что сократит срок доставки заказов | 2.000000 |
| **5** | Добавить страницу отзывов клиентов о магазине, что позволит увеличить количество заказов | 1.333333 |
| **3** | Изменить структура категорий, что увеличит конверсию, т.к. пользователи быстрее найдут нужный товар | 1.125000 |
| **4** | Изменить цвет фона главной страницы, чтобы увеличить вовлеченность пользователей | 1.000000 |

Наиболее перспективные гипотезы 8, 0, 7, 6.

Посчитаем RICE для гипотез. Добавим полученные значения RICE к существующей таблице. Новый столбец назовем RICE. Выведем столбцы 'hypothesis' и 'RICE', отсортированные по столбцу 'RICE' по убыванию приоритета.

Ввод [4]:

hyps['RICE'] **=** (hyps['reach']**\***hyps['impact']**\***hyps['confidence'])**/**hyps['efforts']

riced\_hyps **=** hyps[['hypothesis', 'RICE']].sort\_values('RICE', ascending**=False**)

riced\_hyps

Out[4]:

|  | **hypothesis** | **RICE** |
| --- | --- | --- |
| **7** | Добавить форму подписки на все основные страницы, чтобы собрать базу клиентов для email-рассылок | 112.0 |
| **2** | Добавить блоки рекомендаций товаров на сайт интернет магазина, чтобы повысить конверсию и средний чек заказа | 56.0 |
| **0** | Добавить два новых канала привлечения трафика, что позволит привлекать на 30% больше пользователей | 40.0 |
| **6** | Показать на главной странице баннеры с актуальными акциями и распродажами, чтобы увеличить конверсию | 40.0 |
| **8** | Запустить акцию, дающую скидку на товар в день рождения | 16.2 |
| **3** | Изменить структура категорий, что увеличит конверсию, т.к. пользователи быстрее найдут нужный товар | 9.0 |
| **1** | Запустить собственную службу доставки, что сократит срок доставки заказов | 4.0 |
| **5** | Добавить страницу отзывов клиентов о магазине, что позволит увеличить количество заказов | 4.0 |
| **4** | Изменить цвет фона главной страницы, чтобы увеличить вовлеченность пользователей | 3.0 |

Наиболее перспективные гипотезы 7, 2, 0, 6.

Вывод.

Наиболее перспективными гипотезами при выборе приоритизации через ICE стали:

* Запустить акцию, дающую скидку на товар в день рождения(16 ед);
* Добавить два новых канала привлечения трафика, что позволит привлекать на 30% больше пользователей (13 ед);
* Добавить форму подписки на все основные страницы, чтобы собрать базу клиентов для email-рассылок (11 ед);
* Показать на главной странице баннеры с актуальными акциями и распродажами, чтобы увеличить конверсию (8 ед).

Наиболее перспективными гипотезами при выборе приоритизации через RICE стали:

* Добавить два новых канала привлечения трафика, что позволит привлекать на 30% больше пользователей (112 ед);
* Добавить блоки рекомендаций товаров на сайт интернет магазина, чтобы повысить конверсию и средний чек заказа (56 ед);
* Добавить два новых канала привлечения трафика, что позволит привлекать на 30% больше пользователей (40 ед);
* Показать на главной странице баннеры с актуальными акциями и распродажами, чтобы увеличить конверсию (40 ед).

При смене варианта приоритезации с ICE на RICE гипотеза 8 уступила первое место гипотезе 7 так как уровень охвата пользователей последней гораздо выше (10 ед) при одинаковых затратах ресурсов на их проверку (по 5 ед на каждую из гипотез); по той же причине на второе место вышла гипотеза 2, сменив гипотезу 0 - охват пользователей больше, а затраты на проверку меньше. Гипотеза о привлечении трафика через 2 новых канала переместилась на 3 место из-за более высоких затрат на ее проверку и при этом меньшего охвата пользователей. Гипотеза на 4 месте осталась неизменной. И все благодаря новому параметру reach, который учитывает модель приоритизации RICE, гипотезы, полученные с помощью именно этого фреймворка рекомендуются к проверке в А/В-тесте.

# Анализ A/B-теста.

Ввод [5]:

*#выгрузка и ознакомление с данными датасета с заказами*

orders **=** pd.read\_csv('/datasets/orders.csv')

orders.info()

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

RangeIndex: 1197 entries, 0 to 1196

Data columns (total 5 columns):

# Column Non-Null Count Dtype

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

0 transactionId 1197 non-null int64

1 visitorId 1197 non-null int64

2 date 1197 non-null object

3 revenue 1197 non-null int64

4 group 1197 non-null object

dtypes: int64(3), object(2)

memory usage: 46.9+ KB

В датафрейме orders 1197 строк и некорректный тип данных в столбце date. Пропусков нет.

Ввод [6]:

*#выгрузка и ознакомление с данными датасета с клиентами*

visitors **=** pd.read\_csv('/datasets/visitors.csv')

visitors.info()

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

RangeIndex: 62 entries, 0 to 61

Data columns (total 3 columns):

# Column Non-Null Count Dtype

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

0 date 62 non-null object

1 group 62 non-null object

2 visitors 62 non-null int64

dtypes: int64(1), object(2)

memory usage: 1.6+ KB

В датафрейме visitors 62 строки и некорректный тип данных в столбце date. Пропусков нет.

Ввод [7]:

*# преобразую данные о времени*

orders['date'] **=** pd.to\_datetime(orders['date'])

​

visitors['date'] **=** pd.to\_datetime(visitors['date'])

​

orders.info()

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

RangeIndex: 1197 entries, 0 to 1196

Data columns (total 5 columns):

# Column Non-Null Count Dtype

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

0 transactionId 1197 non-null int64

1 visitorId 1197 non-null int64

2 date 1197 non-null datetime64[ns]

3 revenue 1197 non-null int64

4 group 1197 non-null object

dtypes: datetime64[ns](1), int64(3), object(1)

memory usage: 46.9+ KB

Ввод [8]:

visitors.info()

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

RangeIndex: 62 entries, 0 to 61

Data columns (total 3 columns):

# Column Non-Null Count Dtype

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

0 date 62 non-null datetime64[ns]

1 group 62 non-null object

2 visitors 62 non-null int64

dtypes: datetime64[ns](1), int64(1), object(1)

memory usage: 1.6+ KB

Изменения сохранены.

Ввод [9]:

*# конвертеры, которые позволяют использовать типы pandas в matplotlib*

register\_matplotlib\_converters()

Ввод [24]:

orders.duplicated().sum()

​

Out[24]:

0

Ввод [25]:

visitors.duplicated().sum()

Out[25]:

0

Ввод [10]:

*# создаем массив уникальных пар значений дат и групп теста*

datesGroups **=** orders[['date','group']].drop\_duplicates()

​

*# получаем агрегированные кумулятивные по дням данные о заказах*

ordersAggregated **=** datesGroups.apply(**lambda** x: orders[np.logical\_and(orders['date'] **<=** x['date'], orders['group'] **==** x['group'])].agg({'date' : 'max', 'group' : 'max', 'transactionId' : 'nunique', 'visitorId' : 'nunique', 'revenue' : 'sum'}), axis**=**1).sort\_values(by**=**['date','group'])

​

*# получаем агрегированные кумулятивные по дням данные о посетителях интернет-магазина*

visitorsAggregated **=** datesGroups.apply(**lambda** x: visitors[np.logical\_and(visitors['date'] **<=** x['date'], visitors['group'] **==** x['group'])].agg({'date' : 'max', 'group' : 'max', 'visitors' : 'sum'}), axis**=**1).sort\_values(by**=**['date','group'])

​

*# объединяем кумулятивные данные в одной таблице и присваиваем ее столбцам понятные названия*

cumulativeData **=** ordersAggregated.merge(visitorsAggregated, left\_on**=**['date', 'group'], right\_on**=**['date', 'group'])

cumulativeData.columns **=** ['date', 'group', 'orders', 'buyers', 'revenue', 'visitors']

​

*#построим графики кумулятивной выручки по дням и группам A/B-тестирования*

*# датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням в группе А*

cumulativeRevenueA **=** cumulativeData[cumulativeData['group']**==**'A'][['date','revenue', 'orders']]

​

*# датафрейм с кумулятивным количеством заказов и кумулятивной выручкой по дням в группе B*

cumulativeRevenueB **=** cumulativeData[cumulativeData['group']**==**'B'][['date','revenue', 'orders']]

​

*# Строим график выручки группы А*

plt.plot(cumulativeRevenueA['date'], cumulativeRevenueA['revenue'], label**=**'A')

​

*# Строим график выручки группы B*

plt.plot(cumulativeRevenueB['date'], cumulativeRevenueB['revenue'], label**=**'B')

​

*# размер шрифта по оси x*

plt.tick\_params(axis**=**'x', which**=**'major', labelsize**=**7)

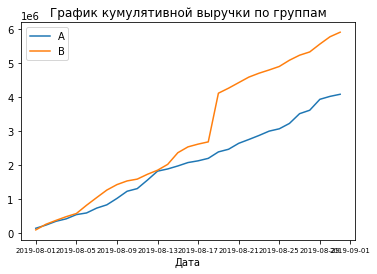
​

plt.title('График кумулятивной выручки по группам')

plt.xlabel('Дата')

plt.legend();

​



Выручка почти равномерно увеличивается в течение всего теста. Однако график выручки группы В в 2 точках резко растет, что может сигнализировать о всплесках числа заказов, либо о появлении очень дорогих заказов в выборке. Примерно в середине теста выручка группы В значительно превосходит выручку по группе А и продолжает расти.

Ввод [11]:

*#построим график кумулятивного среднего чека по группам*

plt.plot(cumulativeRevenueA['date'], cumulativeRevenueA['revenue']**/**cumulativeRevenueA['orders'], label**=**'A')

plt.plot(cumulativeRevenueB['date'], cumulativeRevenueB['revenue']**/**cumulativeRevenueB['orders'], label**=**'B')

*# размер шрифта по оси x*

plt.tick\_params(axis**=**'x', which**=**'major', labelsize**=**7)

​

plt.title('График кумулятивного среднего чека по группам')

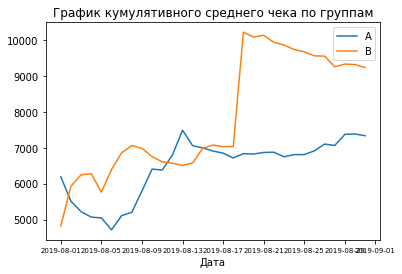
plt.xlabel('Дата')

plt.legend()

​

Out[11]:

<matplotlib.legend.Legend at 0x7f421601b6d0>



Средний чек так же стабилизируется к концу теста для группы А и значительно более высокий для группы В, возможно в группу В в первой половине теста попали крупные заказы, а во второй 1-2 еще более крупных (чуть более плавные 7 августа и совсем резкие скачки 19 августа на графике примерно). Возможно для теста требуется больше данных для того чтобы прийти к реальному среднему чеку и установиться на его уровне.

Ввод [12]:

*#построим график относительного различия для среднего чека*

*# собираем данные в одном датафрейме*

mergedCumulativeRevenue **=** cumulativeRevenueA.merge(cumulativeRevenueB, left\_on**=**'date', right\_on**=**'date', how**=**'left', suffixes**=**['A', 'B'])

​

*# cтроим отношение средних чеков*

plt.plot(mergedCumulativeRevenue['date'], (mergedCumulativeRevenue['revenueB']**/**mergedCumulativeRevenue['ordersB'])**/**(mergedCumulativeRevenue['revenueA']**/**mergedCumulativeRevenue['ordersA'])**-**1)

​

*# добавляем ось X*

plt.axhline(y**=**0, color**=**'black', linestyle**=**'--')

*# размер шрифта по оси x*

plt.tick\_params(axis**=**'x', which**=**'major', labelsize**=**7)

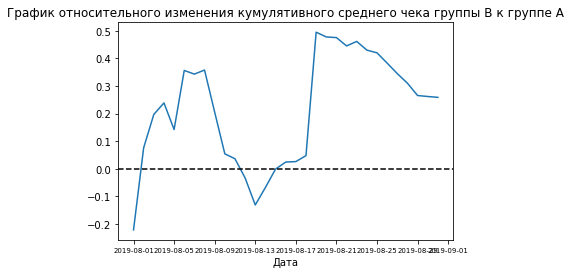
plt.title('График относительного изменения кумулятивного среднего чека группы B к группе A')

plt.xlabel('Дата')

​

Out[12]:

Text(0.5, 0, 'Дата')



Мы можем наблюдать резкие скачки на графике в нескольких точках, что убеждает нас в наличии крупных заказов (выбросов в данных).

Ввод [13]:

*#построим график кумулятивного среднего количества заказов на посетителя по группам*

*# считаем кумулятивную конверсию*

cumulativeData['conversion'] **=** cumulativeData['orders']**/**cumulativeData['visitors']

​

*# отделяем данные по группе A*

cumulativeDataA **=** cumulativeData[cumulativeData['group']**==**'A']

​

*# отделяем данные по группе B*

cumulativeDataB **=** cumulativeData[cumulativeData['group']**==**'B']

​

*# строим графики*

plt.plot(cumulativeDataA['date'], cumulativeDataA['conversion'], label**=**'A')

plt.plot(cumulativeDataB['date'], cumulativeDataB['conversion'], label**=**'B')

plt.legend()

*# размер шрифта по оси x*

plt.tick\_params(axis**=**'x', which**=**'major', labelsize**=**7)

​

*# задаем масштаб осей*

plt.axis([dt.datetime(2019, 8, 1), dt.datetime(2019, 9, 1), 0.025, 0.04])

​

plt.title('График кумулятивной конверсии по группам')

plt.xlabel('Дата')

Out[13]:

Text(0.5, 0, 'Дата')

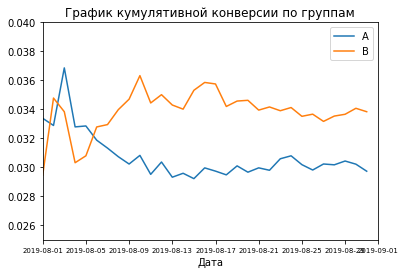


График выглядит немного симметричным после первой недели эксперимента, в конце которой показатели пересеклись и достигли равенства, но далее показатели каждой из групп почти выровнялись и количество заказов группы В стало стабильно высоким с единичными скачками (в дни большого количества заказов), а количество заказов группы А стабильно низким, а к концу эксперимента снижающимся еще больше.

Ввод [14]:

*#построим график относительного изменения кумулятивного среднего количества заказов на посетителя группы B к группе A*

mergedCumulativeConversions **=** cumulativeDataA[['date','conversion']].merge(cumulativeDataB[['date','conversion']], left\_on**=**'date', right\_on**=**'date', how**=**'left', suffixes**=**['A', 'B'])

​

plt.plot(mergedCumulativeConversions['date'], mergedCumulativeConversions['conversionB']**/**mergedCumulativeConversions['conversionA']**-**1, label**=**"Относительный прирост конверсии группы B относительно группы A")

plt.legend()

​

plt.axhline(y**=**0, color**=**'black', linestyle**=**'--')

plt.axhline(y**=**0.2, color**=**'grey', linestyle**=**'--')

plt.axis(["2019-08-01", '2019-09-01', **-**0.3, 0.4])

​

*# размер шрифта по оси x*

plt.tick\_params(axis**=**'x', which**=**'major', labelsize**=**7)

​

plt.title('График относительного изменения кумулятивной конверсии группы B к группе A')

plt.xlabel('Дата')

​

Out[14]:

Text(0.5, 0, 'Дата')



В начале теста группа А опережала группу В по показателю, но затем сдала позицию и проджила плавное падение. Кумулятивная же конверси группы В подскочила и растет в течение всего эксперимента.

Ввод [15]:

*#построим точечный график количества заказов по пользователям*

ordersByUsers **=** (

orders.groupby('visitorId', as\_index**=False**)

.agg({'transactionId': 'nunique'})

)

​

ordersByUsers.columns **=** ['visitorId', 'orders']

x\_values **=** pd.Series(range(0,len(ordersByUsers)))

​

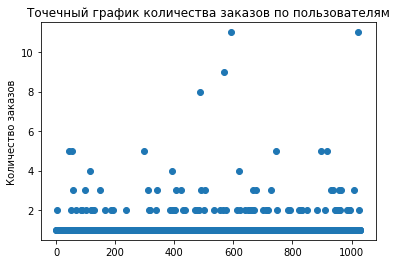
plt.scatter(x\_values, ordersByUsers['orders'])

plt.title('Точечный график количества заказов по пользователям')

plt.ylabel('Количество заказов')

Out[15]:

Text(0, 0.5, 'Количество заказов')



Как мы видим на графике чаще всего пользователь делает 1 или 2 заказа, но не видим их точную долю, необходимо определиться с тем, что считать аномалией.

Ввод [16]:

*#посчитаем 95-й и 99-й перцентили количества заказов на пользователя, выберем границу для определения аномальных пользователей*

print(np.percentile(ordersByUsers['orders'], [95, 99]))

[2. 4.]

Не более 5% пользователей совершали больше двух заказов и не более 1% пользователей - больше 4 заказов. Считаю целесообразным выбрать 2 заказа на одного пользователя за нижнюю границу числа заказов, и отсеять аномальных пользователей по ней.

Ввод [17]:

*#построим точечный график стоимостей заказов по пользователям*

​

x\_values **=** pd.Series(range(0,len(orders['revenue'])))

​

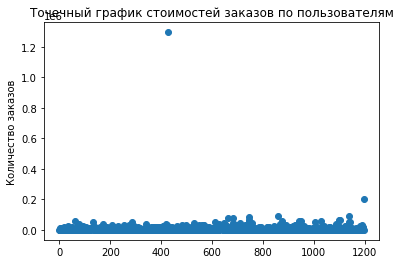
plt.scatter(x\_values, orders['revenue'])

plt.title('Точечный график стоимостей заказов по пользователям')

plt.ylabel('Количество заказов')

Out[17]:

Text(0, 0.5, 'Количество заказов')



На графике мы можем видеть один сверхдорогой заказ по стоимости около 1300тр. и один заказ стоимостью около 200тр. - вот они наши аномалии. Так же можно сделать вывод, что потолок нормальной стоимости заказов находится на высоте 90тр.

Ввод [18]:

*#посчитаем 95-й и 99-й перцентили стоимости заказов на пользователя, выберем границу для определения аномальных пользователей*

print(np.percentile(orders['revenue'], [95, 99]))

[28000. 58233.2]

Не более 5% пользователей совершали покупку более чем на 28тр. и не более 1% пользователей совершали заказ более чем на 58233р. Границей стоит обозначить 28тр.

Ввод [28]:

*#посчитаем статистическую значимость различия в среднем количестве заказов между группами*

*#Сформулируем гипотезы.*

*#Нулевая: различий в среднем количестве заказов между группами нет.*

*#Альтернативная: различия в среднем между группами есть.*

​

visitorsADaily **=** visitors[visitors['group'] **==** 'A'][['date', 'visitors']]

visitorsADaily.columns **=** ['date', 'visitorsPerDateA']

​

visitorsACummulative **=** visitorsADaily.apply(

**lambda** x: visitorsADaily[visitorsADaily['date'] **<=** x['date']].agg(

{'date': 'max', 'visitorsPerDateA': 'sum'}

),

axis**=**1,

)

visitorsACummulative.columns **=** ['date', 'visitorsCummulativeA']

​

visitorsBDaily **=** visitors[visitors['group'] **==** 'B'][['date', 'visitors']]

visitorsBDaily.columns **=** ['date', 'visitorsPerDateB']

​

visitorsBCummulative **=** visitorsBDaily.apply(

**lambda** x: visitorsBDaily[visitorsBDaily['date'] **<=** x['date']].agg(

{'date': 'max', 'visitorsPerDateB': 'sum'}

),

axis**=**1,

)

visitorsBCummulative.columns **=** ['date', 'visitorsCummulativeB']

​

ordersADaily **=** (

orders[orders['group'] **==** 'A'][['date', 'transactionId', 'visitorId', 'revenue']]

.groupby('date', as\_index**=False**)

.agg({'transactionId': pd.Series.nunique, 'revenue': 'sum'})

)

ordersADaily.columns **=** ['date', 'ordersPerDateA', 'revenuePerDateA']

​

ordersACummulative **=** ordersADaily.apply(

**lambda** x: ordersADaily[ordersADaily['date'] **<=** x['date']].agg(

{'date': 'max', 'ordersPerDateA': 'sum', 'revenuePerDateA': 'sum'}

),

axis**=**1,

).sort\_values(by**=**['date'])

ordersACummulative.columns **=** [

'date',

'ordersCummulativeA',

'revenueCummulativeA',

]

ordersBDaily **=** (

orders[orders['group'] **==** 'B'][['date', 'transactionId', 'visitorId', 'revenue']]

.groupby('date', as\_index**=False**)

.agg({'transactionId': pd.Series.nunique, 'revenue': 'sum'})

)

ordersBDaily.columns **=** ['date', 'ordersPerDateB', 'revenuePerDateB']

​

ordersBCummulative **=** ordersBDaily.apply(

**lambda** x: ordersBDaily[ordersBDaily['date'] **<=** x['date']].agg(

{'date': 'max', 'ordersPerDateB': 'sum', 'revenuePerDateB': 'sum'}

),

axis**=**1,

).sort\_values(by**=**['date'])

ordersBCummulative.columns **=** [

'date',

'ordersCummulativeB',

'revenueCummulativeB',

]

data **=** (

ordersADaily.merge(

ordersBDaily, left\_on**=**'date', right\_on**=**'date', how**=**'left'

)

.merge(ordersACummulative, left\_on**=**'date', right\_on**=**'date', how**=**'left')

.merge(ordersBCummulative, left\_on**=**'date', right\_on**=**'date', how**=**'left')

.merge(visitorsADaily, left\_on**=**'date', right\_on**=**'date', how**=**'left')

.merge(visitorsBDaily, left\_on**=**'date', right\_on**=**'date', how**=**'left')

.merge(visitorsACummulative, left\_on**=**'date', right\_on**=**'date', how**=**'left')

.merge(visitorsBCummulative, left\_on**=**'date', right\_on**=**'date', how**=**'left')

)

​

*# для пользователей, совершивших хотя бы 1 заказ, будет указано число заказов*

ordersByUsersA **=** (

orders[orders['group'] **==** 'A']

.groupby('visitorId', as\_index**=False**)

.agg({'transactionId': pd.Series.nunique})

)

ordersByUsersA.columns **=** ['userId', 'orders']

​

ordersByUsersB **=** (

orders[orders['group'] **==** 'B']

.groupby('visitorId', as\_index**=False**)

.agg({'transactionId': pd.Series.nunique})

)

ordersByUsersB.columns **=** ['userId', 'orders']

*# пользователям с заказами будет соответствовать число заказов пользователя, а пользователям без заказов — нули*

sampleA **=** pd.concat([ordersByUsersA['orders'],pd.Series(0, index**=**np.arange(data['visitorsPerDateA'].sum() **-** len(ordersByUsersA['orders'])), name**=**'orders')],axis**=**0)

​

sampleB **=** pd.concat([ordersByUsersB['orders'],pd.Series(0, index**=**np.arange(data['visitorsPerDateB'].sum() **-** len(ordersByUsersB['orders'])), name**=**'orders')],axis**=**0)

​

print("{0:.5f}".format(st.mannwhitneyu(sampleA, sampleB)[1]))

​

print("{0:.3f}".format(sampleB.mean() **/** sampleA.mean() **-** 1))

0.01679

0.138

Р-value = 0.01679 меньше 0.05, следовательно, отвергаем нулевую гипотезу о том, что статистически значимых различий в конверсии между группами нет, по «сырым» данным различия в среднем числе заказов групп A и B есть. Относительный прирост конверсии группы В к конверсии группы А составил 13,8%.

Ввод [20]:

*#теперь проверим статистическую значимость различий в среднем чеке между группами по «сырым» данным.*

*#Нулевая гипотеза: различий в среднем чеке между группами нет.*

*#Альтернативная гипотеза: различия в среднем чеке между группами есть.*

print('{0:.3f}'.format(st.mannwhitneyu(orders[orders['group']**==**'A']['revenue'], orders[orders['group']**==**'B']['revenue'])[1]))

print('{0:.3f}'.format(orders[orders['group']**==**'B']['revenue'].mean()**/**orders[orders['group']**==**'A']['revenue'].mean()**-**1))

​

0.729

0.259

P-value больше 0.05 (равен 0.729). Значит, причин отвергать нулевую гипотезу и считать, что в среднем чеке есть различия, нет. Относительное различие среднего чека между сегментами - 25,9% не является статистически значимым и может исчезнуть после избавления от выбросов.

Ввод [27]:

*#посчитаем статистическую значимость различий в среднем количестве заказов на посетителя между группами по «очищенным» данным*

*#Сформулируем гипотезы.*

*#Нулевая: различий в среднем количестве заказов между группами по очищенным данным нет.*

*#Альтернативная: различия в среднем между группами по очищенным данным есть.*

usersWithManyOrders **=** pd.concat(

[

ordersByUsersA[ordersByUsersA['orders'] **>** 2]['userId'],

ordersByUsersB[ordersByUsersB['orders'] **>** 2]['userId'],

],

axis**=**0,

)

​

*# срезы пользователей*

usersWithExpensiveOrders **=** orders[orders['revenue'] **>** 28000]['visitorId']

abnormalUsers **=** (

pd.concat([usersWithManyOrders, usersWithExpensiveOrders], axis**=**0)

.drop\_duplicates()

.sort\_values()

)

sampleAFiltered **=** pd.concat(

[

ordersByUsersA[

np.logical\_not(ordersByUsersA['userId'].isin(abnormalUsers))

]['orders'],

pd.Series(

0,

index**=**np.arange(

data['visitorsPerDateA'].sum() **-** len(ordersByUsersA['orders'])

),

name**=**'orders',

),

],

axis**=**0,

)

​

sampleBFiltered **=** pd.concat(

[

ordersByUsersB[

np.logical\_not(ordersByUsersB['userId'].isin(abnormalUsers))

]['orders'],

pd.Series(

0,

index**=**np.arange(

data['visitorsPerDateB'].sum() **-** len(ordersByUsersB['orders'])

),

name**=**'orders',

),

],

axis**=**0,

)

​

print('{0:.5f}'.format(st.mannwhitneyu(sampleAFiltered, sampleBFiltered)[1]))

print('{0:.3f}'.format(sampleBFiltered.mean()**/**sampleAFiltered.mean()**-**1))

0.01304

0.173

Мы удалили из данных аномалии в виде пользователей, которые совершили более 2 покупок или имели чек более чем на 28тр. Результаты по среднему количеству заказов практически не изменились, после очиски данных относительная разница между группами составила 17.3%.

Ввод [26]:

*#посчитаем статистическую значимость различий в среднем чеке заказа между группами по «очищенным» данным.*

*#Нулевая гипотеза: различий в среднем чеке между группами по очищенным данным нет.*

*#Альтернативная гипотеза: различия в среднем чеке между группами по очищенным данным есть.*

print(

'{0:.3f}'.format(

st.mannwhitneyu(

orders[

np.logical\_and(

orders['group'] **==** 'A',

np.logical\_not(orders['visitorId'].isin(abnormalUsers)),

)

]['revenue'],

orders[

np.logical\_and(

orders['group'] **==** 'B',

np.logical\_not(orders['visitorId'].isin(abnormalUsers)),

)

]['revenue'],

)[1]

)

)

​

print(

"{0:.3f}".format(

orders[

np.logical\_and(

orders['group'] **==** 'B',

np.logical\_not(orders['visitorId'].isin(abnormalUsers)),

)

]['revenue'].mean()

**/** orders[

np.logical\_and(

orders['group'] **==** 'A',

np.logical\_not(orders['visitorId'].isin(abnormalUsers)),

)

]['revenue'].mean()

**-** 1

)

)

​

0.738

-0.020

P-value все еще больше 0.05 (равен 0.738), увеличился как и разница между сегментами, продолжаем не отвергать нулевую гипотезу. А наблюдаемое различие в 2% не имеет статичтической значимости.

# Вывод и Решение по результатам теста.

В ходе анализа мною были рассмотрены, изучены и преобразованы полученные данные для дальнейшего анализа списка гипотез для увеличения прибыли интернет-магазина. Гипотезы были приоретизированы при помощи фреймворков ICE и RICE, в ходе данного процесса был сделан вывод, что наилучшим и полным фреймворком является последний благодаря большему количеству параметров. Далее был проведен анализ A/B-теста с помощью средств визуализации(графики), проведено сравнение по ключевым показателям по сырым и очищенным от выбросов данным, посчитана статистическая значимость различий для групп А и В по количеству заказов и сумме среднего чека и сделаны следующие выводы:

* график различия конверсий демонстрирует нам лидерство группы В;
* выручка в группе В так же значительно выше и растет;
* после очистки данных различие в среднем чеке составило 2%, что не имеет статистической значимости;
* количество заказов в группе В стабильно выше этого показателя в группе А;
* среднее количество заказов на пользователя равно 2, а средняя сумма чека - 28тр;
* выявлено статистически значимое различие данных по конверсии как после так и до очистки данных;
* конверсия пользователей из группы В на 17,3% лучше конверсии пользователей группы А.

Исходя из полученных выше результатов могу рекомендовать остановить тест и зафиксировать победу за группой В, которая является несомненным лидером по конверсии как до очистки данных, так и после, а превосходство группы А по среднему чеку после очистки данных является незначительным.