Преамбула

> Для данных исследований необходимо три библиотеки **- pandas, numpy, ast.** Эти библиотеки ставятся по умолчанию в юпитере**,** потому не думаю что могут возникнуть проблемы с конфликтом версий**.** ***!!!Внимание!!!*** Проверьте путь к файлу датасета**,** данные импортируются дважды тут и тут**.** Причины ниже**)**

Исследование действий пользователей сайта

> ***Содержание***
>
>

In [1]:

```
# Необходимые библиотеки для исследования
import pandas as pd
import numpy as np
from ast import literal_eval


# формат отображения
pd.options.display.float_format = '{:.1f}'.format

# импорт датасета
df_users = pd.read_csv('data.csv')

# для удобства, все столбцы приводятся к нижнему регистру
df_users.columns = df_users.columns.str.lower()
```

In [2]:

```
def first_look (df):
    '''Функция получения первичной информации о датафрейме'''
    print ('------------- Первые 5 строк ------------')
    display(df.head())
    print('')
    print('')
    print ('------------- Типы данных ------------')
    print (df.info())
    print('')
    print('')
    print ('------------- Пропуски ------------')
    count = 0
    for element in df.columns:
        if df[element].isna().sum() > 0:
            print(element, ' - ', df[element].isna().sum(), 'пропусков')
            count = +1
        if count == 0:
            print('Пропусков НЕТ')
```

```
    print('')
    print('')
```

In [3]:

```
first_look(df_users)
```

```
------------ Первые 5 строк ------------
```

	unnamed: 0	clientid	url	goalsid	paramskeys
0	0	1543250134295194505	https://logiclike.com/	NaN	['cardId','region','userId','registra
1	1	1540396567213978260	https://logiclike.com/	NaN	['cardId','region','userId','registra
2	2	1540396567213978260	https://logiclike.com/	NaN	
3	3	1540396567213978260	https://logiclike.com/user#/quiz/73985/process	NaN	
4	4	1540396567213978260	https://logiclike.com/cabinet#/quiz/73985/process	NaN	

```
------------ Типы данных ------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 306383 entries, 0 to 306382
Data columns (total 6 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   unnamed: 0     306383 non-null  int64  
 1   clientid       306383 non-null  uint64 
 2   url            306383 non-null  object 
 3   goalsid        6627 non-null    float64
 4   paramskeys     306383 non-null  object 
 5   paramsvalues   306383 non-null  object 
dtypes: float64(1), int64(1), object(3), uint64(1)
memory usage: 14.0+ MB
None


------------ Пропуски ------------
Пропусков НЕТ
Пропусков НЕТ
Пропусков НЕТ
goalsid  -  299756 пропусков
```

> На первый взгляд все данные корректны, в **goalsid** имеется большое количество пропусков, есть лишний столбец, дублирующий индексы. Это нормально. Ликвидировать их не стоит. Проверка столбца **paramskeys** на корректность.

In [4]:

```
# просмотр уникальных значений столбца
df_users['paramskeys'].unique()
```

Out[4]:

```
array(["['cardId','region','userId','registrationType','targetLessons']",
       "['ip']", '[]', "['targetLessons']",
       "['targetLessons','cardId','userId','registrationType']",
       "['cardId','userId','registrationType']"], dtype=object)
```

In [5]:

```
s="['cardId','region','userId','registrationType','targetLessons']"
eval(s)
```

Out[5]:

```
['cardId', 'region', 'userId', 'registrationType', 'targetLessons']
```

In [6]:

```
df_users['paramskeys'] = df_users['paramskeys'].apply(eval)
df_users
```

Out[6]:

	unnamed: 0	clientid	url	goalsid	paramskeys	para
0	0	1543250134295194505	https://logiclike.com/	NaN	[cardId, region, userId, registrationType, tar...	['215
1	1	1540396567213978260	https://logiclike.com/	NaN	[cardId, region, userId, registrationType, tar...	['39
2	2	1540396567213978260	https://logiclike.com/	NaN	[ip]	
3	3	1540396567213978260	https://logiclike.com/user#/quiz/73985/process	NaN	[]	
4	4	1540396567213978260	https://logiclike.com/cabinet#/quiz/73985/process	NaN	[ip]	
...	...	...	...	...	...	
306378	306378	1547201659806207786	https://logiclike.com/cabinet#/quiz/73429/result	NaN	[cardId, region, userId, registrationType, tar...	['87
306379	306379	1547211249801412828	https://logiclike.com/	NaN	[ip]	
306380	306380	1547211249801412828	https://logiclike.com/	NaN	[cardId, region, userId, registrationType, tar...	['87
306381	306381	1547211249801412828	https://logiclike.com/	25230759.0	[]	
306382	306382	1546426814607550 30	https://logiclike.com/cabinet#/quiz/75007/process	NaN	[]	

306383 rows × 6 columns

> Найдено несколько аномалий
>
> 1. Содержимое ячеек не является списками - это строки.
> 2. Параметр **targetLessons** может находится в любой части строки, потому можно заключить, что соотвествующее ему значение тоже может находится в разных "местах" строки.
>
> Файл нужно переимпортировать с соответсвующими параметрами, превращая строчные значения **paramskeys** и **paramsvalues** в списки. Можно было бы воспользоваться методом ***eval*** но мне не нравится время его работы, он долгий, переимпортировние файла занимает в два раза меньше времени.

In [7]:

```
# переимпортирование - удаление лишнего столбца, перевод строчных значений в списочные в нужных столбцах
df_users = pd.read_csv('data.csv', index_col = 0, converters = {'paramsKeys':literal_eval, 'paramsValues':literal_eval})
# для удобства, все столбцы приводятся к нижнему регистру
df_users.columns = df_users.columns.str.lower()
```

In [8]:

In [8]:

```
df_users
```

Out[8]:

	clientid	url	goalsid	paramskeys	paramsvalues
0	1543250134295194505	https://logiclike.com/	NaN	[cardId, region, userId, registrationType, tar...	[215499, Россия, 221529, null, 2]
1	1540396567213978260	https://logiclike.com/	NaN	[cardId, region, userId, registrationType, tar...	[394451, Минск, 340628, null, 0]
2	1540396567213978260	https://logiclike.com/	NaN	[ip]	[178.120.38.195]
3	1540396567213978260	https://logiclike.com/user#/quiz/73985/process	NaN	[]	[]
4	1540396567213978260	https://logiclike.com/cabinet#/quiz/73985/process	NaN	[ip]	[178.120.38.195]
...	...	...	...	...	...
306378	1547201659806207786	https://logiclike.com/cabinet#/quiz/73429/result	NaN	[cardId, region, userId, registrationType, tar...	[877460, Россия, 880366, 18, 2]
306379	1547211249801412828	https://logiclike.com/	NaN	[ip]	[85.141.127.134]
306380	1547211249801412828	https://logiclike.com/	NaN	[cardId, region, userId, registrationType, tar...	[878249, Россия, 881170, 18, 2]
306381	1547211249801412828	https://logiclike.com/	25230759.0	[]	[]
306382	154642681460755030	https://logiclike.com/cabinet#/quiz/75007/process	NaN	[]	[]

306383 rows × 5 columns

> Для формирования таблицы с информацией уникльных пользователей, необходимо расчитать систему маркеров. Следующим пунктом будет формирование столбцов с логической меткой действий пользователя, в зависимости от метки в столбце **url**. Будет сформировано **3** столбца ***quiz***, ***process***, ***result***, где маркер, равный **1** означает выполнение действия, пропуск - невыполнения.

In [9]:

```
def extraction_actions (string):
    ''' Функция на входе берет строку и возвращает 1 или nan в зависимости от условия'''
    for action in action_list:
        if action in str(string).split('/'):
            return 1
        else: return np.nan
```

In [10]:

```
# формирование столбца quiz с маркерами
action_list = ['quiz']
df_users['quiz'] = df_users['url'].apply(extraction_actions)
```

In [11]:

```
# формирование столбца process с маркерами
action_list = ['process']
df_users['process'] = df_users['url'].apply(extraction_actions)
```

In [12]:

```
# формирование столбца result с маркерами
action_list = ['result']
df_users['result'] = df_users['url'].apply(extraction_actions)
```

In [13]:

```
df_users.head(5)
```

Out[13]:

	clientid	url	goalsid	paramskeys	paramsvalues	quiz	pl
0	1543250134295194505	https://logiclike.com/	NaN	[cardId, region, userId, registrationType, tar...	[215499, Россия, 221529, null, 2]	NaN	
1	1540396567213978260	https://logiclike.com/	NaN	[cardId, region, userId, registrationType, tar...	[394451, Минск, 340628, null, 0]	NaN	
2	1540396567213978260	https://logiclike.com/	NaN	[ip]	[178.120.38.195]	NaN	
3	1540396567213978260	https://logiclike.com/user#/quiz/73985/process	NaN	[]	[]	1.0	
4	1540396567213978260	https://logiclike.com/cabinet#/quiz/73985/process	NaN	[ip]	[178.120.38.195]	1.0	

> Маркеры сформированы, необходимо сформировать столбец с возрастной группой.

In [14]:

```
def age_category(df):
    '''функция на вход получает датасет
    формирует словарь, для каждой пары paramskeys и paramsvalues, возвращает значение
    соответсвующее targetLessons, а если происходит неудача - возвращает nan'''
    try:
        return dict(zip(df['paramskeys'], df['paramsvalues']))['targetLessons']
    except:
        return np.nan
```

In [15]:

```
df_users['age_category'] = df_users.apply(age_category, axis =1)
```

In [16]:

```
df_users.head(5)
```

Out[16]:

	clientid	url	goalsid	paramskeys	paramsvalues	quiz	pl
0	1543250134295194505	https://logiclike.com/	NaN	[cardId, region, userId, registrationType, tar...	[215499, Россия, 221529, null, 2]	NaN	
1	1540396567213978260	https://logiclike.com/	NaN	[cardId, region, userId, registrationType, tar...	[394451, Минск, 340628, null, 0]	NaN	
2	1540396567213978260	https://logiclike.com/	NaN	[ip]	[178.120.38.195]	NaN	
3	1540396567213978260	https://logiclike.com/user#/quiz/73985/process	NaN	[]	[]	1.0	
4	1540396567213978260	https://logiclike.com/cabinet#/quiz/73985/process	NaN	[ip]	[178.120.38.195]	1.0	

> Маркеры сформированы, необходимо сформировать столбец с уникальными задачами - для подсчета не возможно использовать **URL** поскольку https://logiclike.com/user#/quiz/73985/process и https://logiclike.com/cabinet#/quiz/73985/process - это одна и та же задача (номера одинаковы). Таким образом необходим столбец только с номерами задач, для дальнейших расчетов.

P.S. Я обратил внимание, что в **url** иногда номер задачи содержит букву, типа ***p132567***. Тпких значений около **180**. Ничего с ними делать не стал, поскольку, я не знаю специфики работы сайта, хотя мог бы спокойно "вырвать" эти значения при помощи регулярок)

In [17]:

```
def str_to_number(string):
    '''функция разбивает строку и ищет в ней число, возвращает его'''
    word_list = string.split('/')
    for word in word_list:
        if word.isnumeric():
            return(int(word))
```

In [18]:

```
# нужно присвоить номер только тем задачам, которые решены
df_users['unique_task'] = df_users.loc[df_users['result'].notna(), 'url'].apply(str_to_nu
mber)
```

> Маркеры сформированы, необходимо сформировать столбцы маркеров по действиям "первый вход" и "смотрел тарифы".

In [19]:

```
# лямбда берет на вход значение, и возвращает 1 или пропуск в зависимости от условия
df_users['first_visits'] = df_users['goalsid'].apply(lambda x: 1 if x==25230759 else np.
nan)
df_users['looked'] = df_users['goalsid'].apply(lambda x: 1 if x==40343059 else np.nan)
```

> Маркеры сформированы, предварительная таблица сформирована.
>
> Перед формированием таблицы уникальных пользователей, нужна ликвидация пропусков в столбце **age_category**. Теоретически, для одного и того же пользователя может быть уникальная возрастная группа, либо пропуск. Если высчитать среднее возрастной группы по каждому пользователю, можно получить целое число (при условии что в данных нет ошибки, и одному и тому же пользователю присвоены одинаковые группы). Проверка группировки по клиенту со средними значениями покажет картину)

In [20]:

```
df_users['age_category'] = pd.to_numeric(df_users['age_category'])
d_list = [0,1,2,3]
incorrect = df_users.groupby('clientid')[['age_category']].mean().query('age_category!=@
d_list')
display(incorrect)
display(df_users['clientid'].nunique())
```

	age_category
clientid	
153357826887818237	**0.3**

clientid	age_category
1544793376619875**44**	2.9
1545471045861119**77**	1.7
1546426814607550**30**	0.5
154697345339484784	1.9
...	...
1546870505103176**4929**	0.1
1547044484103791**0829**	1.4
1547201948102309**9888**	1.5
1547220439105645**6463**	2.0
1547238447102870**7916**	1.6

151 rows × 1 columns

```
8490
```

> Ошибочных пользователей имеется **151** штуки**.** Это совсем мизерное количество **-** необходимо удалить**.**

In [21]:

```
# формирование списка некорректых пользователей
incorrect_list = list(incorrect.index)
# новый даасет без некорректных пользователей
df_users = df_users.query('clientid ! = @incorrect_list')
```

In [22]:

```
# в зависимотси от категории (clientid) меняем пропуски на максимальное значение
df_users['age_category'] = df_users.groupby('clientid')['age_category'].apply(lambda x:
x.fillna(x.mean())).astype('int')
```

```
C:\Users\harmf\anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
```

In [23]:

```
# проверка
df_users['age_category'].unique()
```

Out[23]:

```
array([2, 0, 3, 1])
```

> Формирование агрегированной таблицы уникальных пользователей

In [24]:

```
df_users_unique = df_users.groupby('clientid',as_index = False).agg( # группировка по кли
енту
    {'quiz':'sum', # формирование суммы количества переходов на страницу с решением задач
и
     'process':'sum', # сумма просмотренных задач
     'result':'sum',  # сумма решенных задач задач
```

```
     'first_visits':'sum', #
     'looked':'sum', # сумма просмотров тарифов
     'age_category':'mean',
     'unique_task':'nunique'
    })# максимальное по возрастной категории, что и будет возрастной категорией

# Приведение всех параметров к целочисленным значениям
df_users_unique = df_users_unique.astype(int)
```

In [25]:

```
# результат, таблица сформирована
df_users_unique.head(5)
```

Out[25]:

	clientid	quiz	process	result	first_visits	looked	age_category	unique_task
0	**1211105198**	**54**	**18**	**36**	**1**	**0**	**1**	**18**
1	**-1742534562**	**2**	**0**	**2**	**0**	**0**	**1**	**1**
2	**196592066**	**0**	**0**	**0**	**0**	**1**	**1**	**0**
3	**-587538544**	**0**	**0**	**0**	**0**	**1**	**1**	**0**
4	**670416023**	**75**	**25**	**50**	**1**	**0**	**1**	**25**

In [26]:

```
df_users_unique['looked'].unique()
```

Out[26]:

```
array([0, 1])
```

> Наведение "красоты" для формирования итоговой таблицы

In [27]:

```
df_users_unique.columns = \
['Клиенты', 'Действие', 'Задача', 'Результат', 'Первый визит', 'Смотрел тарифы', 'Возраст
ная категория', 'Уникальные задачи']
df_users_unique['Первый визит'] = df_users_unique['Первый визит'].replace([0,1],['Старые
','Новые'])
```

> Формирование итоговой таблицы

In [28]:

```
df_pivot_table = df_users_unique.groupby(['Возрастная категория', 'Первый визит']).agg(
    {'Клиенты':'count', # подсчет числа клиентов
     'Смотрел тарифы':lambda x: '{:.2%}'.format(x.mean()),# формирование доли "смотрел та
рифы"
     'Уникальные задачи':'mean'} # среднее число уникальных задач

)
```

In [31]:

```
df_pivot_table.T
```

Out[31]:

Возрастная категория	0	1	2	3

Возрастная категория	0		1		2		3	
Первый визит	Новые	Старые	Новые	Старые	Новые	Старые	Новые	Старые
Клиенты	39	1133	1678	1472	1267	1254	1273	225
Смотрел тарифы	53.85%	25.68%	25.86%	41.51%	18.07%	38.44%	9.43%	30.22%
Уникальные задачи	11.9	10.8	14.7	10.5	11.1	8.3	6.9	5.1

In [34]:

```
df_users.describe()
```

Out[34]:

	clientid	goalsid	quiz	process	result	age_category	unique_task	first_visits	looked
count	295855.0	6513.0	275238.0	106909.0	168317.0	295855.0	168147.0	4257.0	2256.0
mean	2384836570809735168.0	30465420.3	1.0	1.0	1.0	1.4	74228.6	1.0	1.0
std	3627605008239333376.0	7191247.0	0.0	0.0	0.0	0.9	1290.4	0.0	0.0
min	154723112976302.0	25230759.0	1.0	1.0	1.0	0.0	65382.0	1.0	1.0
25%	1542080031276024832.0	25230759.0	1.0	1.0	1.0	1.0	73355.0	1.0	1.0
50%	1547151942675553024.0	25230759.0	1.0	1.0	1.0	1.0	74703.0	1.0	1.0
75%	1547217768145402368.0	40343059.0	1.0	1.0	1.0	2.0	75472.0	1.0	1.0
max	1547285640107265843 2.0	40343059.0	1.0	1.0	1.0	3.0	75765.0	1.0	1.0

Вывод:

Стоит обратить внимание на первую возрастную категорию.

1. Эта категория насчитывает больше всего пользователей, как новых, так и старых.
2. Среднее число выполненных задач в этой категории либо на уровне с остальными возрастными категориями (новые клиенты показывают около **10 %**), либо превышает этот показатель в остальных категориях (Старые пользователи показывают самый высокий результат в **14,7%**) Высокий параметр в этой возрастной группе говорит о заинтересованности пользователя, а значит позволяет говорить о большей вероятности оформления платной подписки.
3. Процент пользователей смотревших тарифы - высокий у новых пользователей, и относительно небольшой (по сравнению с другими можно считать средний) у старых пользователей, но с учетом того, что пользователей в этой группе много, можно утверждать, что первая возрастная группа - самая преспективная.

Можно обратить внимание и на нулевую - она показывает неплохий показатели, при большом количестве пользователей. В нулевой группе видны высокие результаты у новых пользователей, но это нельзя считать стабильным результатом, поскольку пользователей всего **39**. Можно уведичить время исследований, для подтверждения (либо опровержения) данных показателей. Третья группа, показывает слабые результаты.

Рекомендации:

1. Самой перспективной считаю первыую возрастную группу.
2. Стоит уделить внимание на вторую возрастную группу, поскольку показатели достаточно высокие.
3. Стоит исследовать показатели нулевой и третьей группы, где пользователи показывают высокие результаты, при малом количестве пользователей. Увеличение времени исследования даст возможность оценить, являются ли результаты случайностью, либо имеет место высокий потенциал.

Бонус, функция для построения визуализаций к третьему проекту)

In [ ]:

```
def mean_rel_dev_hist(data, data2):
    tariff_list=['"Смарт"','"Ультра"']
    mn = [
```

```
        'December',
        'November',
        'October',
        'September',
        'August',
        'July',
        'June',
        'May',
        'April',
        'March',
        'February',
        'January'
    ]
    plot_list = ['mb_count', 'messages_count', 'calls_min']
    colors_list = [['#86bf91', 'orange'], ['purple', 'turquoise'], ['red', 'orange']]
    title_list = [
        'Среднее количество Mb трафика в месяц',
        'Среднее количество сообщений в месяц',
        'Среднее кол-во минут разговора в месяц'
    ]
    title_list_hist = [
        'Распределение Mb трафика по пользователям',
        'Распределение количества сообщений по пользователям',
        'Распределение кол-ва минут разговоров по пользователям'
    ]
    border_list = [[15360, 30720], [50, 1000], [500, 3000]]
    df_list = [data, data2]
    hight = [-0.07, 0.3]

    for i in range(len(plot_list)):
        fig, axes = plt.subplots(figsize=(15, 8))
        fig1, axes1 = plt.subplots(figsize=(15.7, 6))

        for j in range(len(df_list)):

            df = df_list[j].pivot_table(index='month', values=plot_list[i])
            df = df.reindex(mn)
            df[plot_list[i]] = df[plot_list[i]].astype(int)
            expenses_dict = dict(df[plot_list[i]])
            label = (
                '\nСреднее - {:.2f}\nДисперсия - {:.2f}\nstd - {:.2f}'
                .format(df_list[j][plot_list[i]].mean(),
                        df_list[j][plot_list[i]].var(ddof=1),
                        df_list[j][plot_list[i]].std(ddof=1)
                       )
            )
            plot = df.plot(
                y=plot_list[i],
                kind='barh',
                ec='black',
                ax=axes,
                width=0.4,
                position = j,
                title=title_list[i],
                color=colors_list[i][j],
                label='Тариф ' + tariff_list[j]
            )
            h=hight[j]
            k = 0
            axes.set_ylim(ymin=-0.5, ymax=11.5)
            for key in expenses_dict:
                if i == 0:
                    mb = str(expenses_dict[key]) + ' mb'
                    plot.text(
                        expenses_dict[key]- 1880,
                        k-h,
                        mb,
                        color = 'white',
                        fontsize='large',
                        fontweight='bold'
                    )
                elif i == 1:
```

```
                plot.text(
                    expenses_dict[key] - 1.5,
                    k-h,
                    expenses_dict[key],
                    color = 'white',
                    fontsize='large',
                    fontweight='bold'
                )
            else:
                min = str(expenses_dict[key]) + ' min'
                plot.text(
                    expenses_dict[key] - 45,
                    k-h,
                    min,
                    color = 'white',
                    fontsize='large',
                    fontweight='bold'
                )
            k += 1
        hist = df_list[j][plot_list[i]].plot(
            kind='hist',
            bins=40,
            ax=axes1,
            alpha=0.5,
            ec='black',
            color = colors_list[i][j],
            label='Тариф ' + tariff_list[j] + label,
            title = title_list_hist[j]
        )
        plt.axvline(
            x=df_list[j][plot_list[i]].median(),
            linewidth=3,
            color=colors_list[i][j],
            linestyle='--',
            label='Медиана ' + tariff_list[j]
        )
        plt.axvline(
            x=df_list[j][plot_list[i]].mean(),
            linewidth=3,
            color=colors_list[i][j],
            linestyle='dotted',
            label='Среднее ' + tariff_list[j]
        )
        plt.axvline(
            x=border_list[i][j],
            linewidth=4,
            color=colors_list[i][j],
            linestyle='solid',
            label='Лимит ' + tariff_list[j]
        )
        axes1.set_xlim(xmin=0)
        plt.legend(loc='upper right')
```