In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import datetime as dt
from scipy.stats import ttest_ind
import math
```

In [2]:
```
df = pd.read_csv('исходник.csv', sep=';')
df
```

Out[2]:

	id_client	flag_conv
0	1345321	0
1	1345322	0
2	1345330	0
3	1345338	1
4	1345342	0
...	...	...
3164	1361073	1
3165	1361074	1
3166	1361082	1
3167	1361083	1
3168	1361088	1

3169 rows × 2 columns

In [3]:
```
df['rs']=df['id_client'].astype(str).str[-1]
df
```

Out[3]:

	id_client	flag_conv	rs
0	1345321	0	1
1	1345322	0	2
2	1345330	0	0
3	1345338	1	8

In [3]:
```
df['rs']=df['id_client'].astype(str).str[-1]
df
```

Out[3]:

	id_client	flag_conv	rs
0	1345321	0	1
1	1345322	0	2
2	1345330	0	0
3	1345338	1	8
4	1345342	0	2
...	...	...	...
3164	1361073	1	3
3165	1361074	1	4
3166	1361082	1	2
3167	1361083	1	3
3168	1361088	1	8

3169 rows × 3 columns

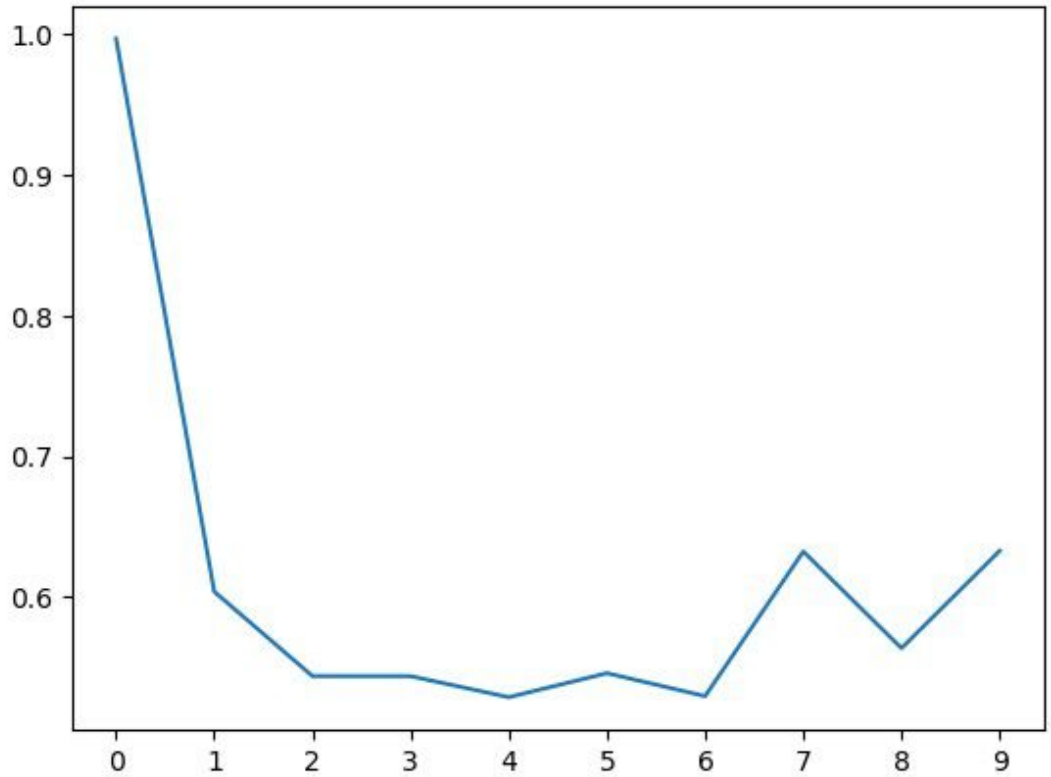
In [4]:
```
df_gr=df.groupby('rs').mean()['flag_conv'].reset_index()
df_gr
```

Out[4]:

	rs	flag_conv
0	0	0.996835
1	1	0.603715
2	2	0.543689
3	3	0.543624
4	4	0.528662
5	5	0.545732
6	6	0.529412
7	7	0.632258
8	8	0.563518
9	9	0.632716

In [5]:
```
plt.plot(df_gr['rs'],df_gr['flag_conv'])
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x216b6db4dc0>]
```



In [6]:
```
# сплит по остатку / на 10

df['id_group_r'] = np.where(df['id_client']%10 < 5, 1, 0)
df
```

Out[6]:

	id_client	flag_conv	rs	id_group_r
0	1345321	0	1	1
1	1345322	0	2	1
2	1345330	0	0	1
3	1345338	1	8	0
4	1345342	0	2	1

3169 rows × 4 columns

In [7]:
```
print(df.groupby('id_group_r').count()['id_client'].reset_index())
print(df.groupby('id_group_r').mean()['flag_conv'].reset_index())
```

```
   id_group_r  id_client
0           0       1609
1           1       1560
   id_group_r  flag_conv
0           0   0.579863
1           1   0.644872
```

```
C:\Users\дом\AppData\Local\Temp\ipykernel_3044\148692164.py:2: FutureWarning: The default value of numeric_only in DataFrameGro
upBy.mean is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only co
lumns which should be valid for the function.
  print(df.groupby('id_group_r').mean()['flag_conv'].reset_index())
```

In [8]:
```
s, p = ttest_ind(df[df['id_group_r']== 1]['flag_conv'], df[df['id_group_r']== 0]['flag_conv'])
print(s)
print(p)
```

```
3.761500955083385
0.00017195329628373794
```

In [9]:
```
#H0  - группы A и B покажут одинаковые конверсии, H1 - не покажут
# s>1.96 - гипотеза H0 вне тела графика
# p<0.05 сила гипотезы H0 слаба
# Следовательно H1 побеждает
# так сплитовать нельзя
```

In [10]:
```
# сплит по остатку / на 2

df['id_group_y'] = np.where(df['id_client']%2 == 0, 1, 0)
df
```

Out[10]:

	id_client	flag_conv	rs	id_group_r	id_group_y
0	1345321	0	1	1	0
1	1345322	0	2	1	1
2	1345330	0	0	1	1
3	1345338	1	8	0	1
4	1345342	0	2	1	1
...	...	...	...	...	...

In [11]:
```
print(df.groupby('id_group_y').count()['id_client'].reset_index())
print(df.groupby('id_group_y').mean()['flag_conv'].reset_index())
```

```
   id_group_y  id_client
0           0       1583
1           1       1586
   id_group_y  flag_conv
0           0   0.591914
1           1   0.631778
```

```
C:\Users\дом\AppData\Local\Temp\ipykernel_3044\2108506452.py:2: FutureWarning: The default value of numeric_only in DataFrameGr
oupBy.mean is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only c
olumns which should be valid for the function.
  print(df.groupby('id_group_y').mean()['flag_conv'].reset_index())
```

In [12]:
```
s, p = ttest_ind(df[df['id_group_y']== 1]['flag_conv'], df[df['id_group_y']== 0]['flag_conv'])
print(s)
print(p)
```

```
2.303661902563631
0.021306090686166332
```

In [13]:
```
#H0  - группы A и B покажут одинаковые конверсии, H1 - не покажут
# s>1.96 - гипотеза H0 вне тела графика
# p<0.05 сила гипотезы H0 слаба
# Следовательно H1 побеждает
# так сплитовать нельзя
```

In [ ]:
```

```

In [14]:
```
# рандомное сплитование

df_aa=pd.DataFrame(columns=['iter', 'stat_crit','p_value'])

for i in range (1,1000):
    f_50=df.sample(frac=0.5)
    s_50 = df.drop(f_50.index)
    s_a, p_a= ttest_ind(f_50['flag_conv'], s_50['flag_conv'])
    df_aa=df_aa.append({'iter':i, 'stat_crit':s_a, 'p_value':p_a }, ignore_index = True)
df_aa
```

In [15]:
```
# % расхождений
ratt = df_aa[df_aa['p_value'] <= 0.05].count()['iter']/df_aa.count()['iter']
print(ratt)
```

```
0.055055055055055056
```

In [16]:
```
# проведем z- =тест
z = (ratt-0.05)/math.sqrt(ratt*(1-ratt)/1000)
print(z)
print(abs(z)<=1.96)
```

```
0.700848319337378
True
```

In [17]:
```
#Сплит можно использовать
```

In [18]:
```
# найдем необходимое количество наблюдений:
```

In [19]:
```
m=df['flag_conv'].mean()
sigma= m*(1-m)
print(m)
print(sigma)
```

```
0.6118649416219628
0.23748623483591486
```

In [20]:
```
n=16*sigma**2/(0.02**2)
print(n)
```

```
2255.988469461572
```