Analyze A/B Test Results

Table of Contents

Part I - Probability

To get started, let's import our libraries.

```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
%matplotlib inline
#We are setting the seed to assure you get the same answers on quizzes
as we set up
random.seed(42)
```

1. Now, read in the ab_data.csv data. Store it in df. **Use your dataframe to answer the questions in Quiz 1 of the classroom.**

a. Read in the dataset and take a look at the top few rows here:

```
#firt we will import the data from csv file
df=pd.read_csv("ab_data.csv")
#showing first 10 rows
df.head(10)
```

```
   user_id                   timestamp      group landing_page
converted
0   851104  2017-01-21 22:11:48.556739    control     old_page
0
1   804228  2017-01-12 08:01:45.159739    control     old_page
0
2   661590  2017-01-11 16:55:06.154213  treatment     new_page
0
3   853541  2017-01-08 18:28:03.143765  treatment     new_page
0
4   864975  2017-01-21 01:52:26.210827    control     old_page
1
5   936923  2017-01-10 15:20:49.083499    control     old_page
0
6   679687  2017-01-19 03:26:46.940749  treatment     new_page
1
```

```
7   719014  2017-01-17 01:48:29.539573    control     old_page
0
8   817355  2017-01-04 17:58:08.979471  treatment     new_page
1
9   839785  2017-01-15 18:11:06.610965  treatment     new_page
1
```

b. Use the below cell to find the number of rows in the dataset.

```
#finding the number of rows in data
df.shape[0]
```

```
294478
```

c. The number of unique users in the dataset.

```
#find how many unique users
df.user_id.nunique()
```

```
290584
```

d. The proportion of users converted.

```
#the porportion of users converted
df['converted'].mean()
```

```
0.11965919355605512
```

e. The number of times the `new_page` and `treatment` don't line up.

```
#we will metion new_page as a
#and mention treatment as b
a_with_no_b=df.query("landing_page == 'new_page' & group != 
'treatment'")
b_with_no_a=df.query("landing_page != 'new_page' & group == 
'treatment'")
#print(len(a_with_no_b))
#print(len(b_with_no_a))
len(a_with_no_b) + len(b_with_no_a)
```

```
3893
```

f. Do any of the rows have missing values?

```
#check if ew have missing values
df.isnull().sum().any()
```

```
False
```

`2.` For the rows where **treatment** is not aligned with **new_page** or **control** is not aligned with **old_page**, we cannot be sure if this row truly received the new or old page. Use **Quiz 2** in the classroom to provide how we should handle these rows.

a. Now use the answer to the quiz to create a new dataset that meets the specifications from the quiz. Store your new dataframe in **df2**.

```
df1 = df.drop(df[(df.group =="treatment") & (df.landing_page !=
"new_page")].index)

df2 = df1.drop(df1[(df.group =="control") & (df1.landing_page !=
"old_page")].index)
```

```
<ipython-input-256-fd0a62ac880b>:3: UserWarning: Boolean Series key
will be reindexed to match DataFrame index.
  df2 = df1.drop(df1[(df.group =="control") & (df1.landing_page !=
"old_page")].index)
```

```
df2[((df2['group'] == 'treatment') == (df2['landing_page'] ==
'new_page')) == False].shape[0]
```

```
0
```

3. Use **df2** and the cells below to answer questions for **Quiz3** in the classroom.

a. How many unique **user_id**s are in **df2**?

```
df2.user_id.nunique()
```

```
290584
```

b. There is one **user_id** repeated in **df2**. What is it?

```
#There is one user_id repeated in df2
df2.user_id.duplicated().sum()
#What is it
df2[df2['user_id'].duplicated()==True]['user_id']
```

```
2893    773192
Name: user_id, dtype: int64
```

c. What is the row information for the repeat **user_id**?

```
#entire row information about duplicated row
df2[df2.duplicated('user_id',keep =False)]
```

```
      user_id                   timestamp      group landing_page
converted
1899   773192  2017-01-09 05:37:58.781806  treatment     new_page
0
2893   773192  2017-01-14 02:55:59.590927  treatment     new_page
0
```

d. Remove **one** of the rows with a duplicate **user_id**, but keep your dataframe as **df2**.

```
#remove one of tow duplicated rows
df2.drop(labels=2893,inplace=True)
```

```
#checking if the row removed or no
df2[df2.duplicated('user_id',keep =False)].shape[0]
```

```
0
```

4. Use **df2** in the below cells to answer the quiz questions related to **Quiz 4** in the classroom.

a. What is the probability of an individual converting regardless of the page they receive?

```
#poportion of converted in new data
df2['converted'].mean()
```

```
0.11959708724499628
```

b. Given that an individual was in the `control` group, what is the probability they converted?

```
#porpotinon of converted which its group is control
df[df['group']=='control']['converted'].mean()
```

```
0.12039917935897611
```

c. Given that an individual was in the `treatment` group, what is the probability they converted?

```
#porpotinon of converted which its group is treatment
df[df['group']=='treatment']['converted'].mean()
```

```
0.11891957956489856
```

d. What is the probability that an individual received the new page?

```
#porpotinon of landing page is new page
len(df[df['landing_page']=='new_page'])/len(df['landing_page'])
```

```
0.5
```

e. Consider your results from a. through d. above, and explain below whether you think there is sufficient evidence to say that the new treatment page leads to more conversions.

> No, there is no sufficient evidence
>
> The test seems to be well designed. Half of the population received the old_page and half of the population received the new_page.
>
> 12.04% that received the old_page were converted. 11.89% that received the new_page were converted. In conclusion, the new_page did not increase the conversion rate.

Part II - A/B Test

Notice that because of the time stamp associated with each event, you could technically run a hypothesis test continuously as each observation was observed.

However, then the hard question is do you stop as soon as one page is considered significantly better than another or does it need to happen consistently for a certain amount of time? How long do you run to render a decision that neither page is better than another?

These questions are the difficult parts associated with A/B tests in general.

`1.` For now, consider you need to make the decision just based on all the data provided. If you want to assume that the old page is better unless the new page proves to be definitely better at a Type I error rate of 5%, what should your null and alternative hypotheses be? You can state your hypothesis in terms of words or in terms of p_{old} and p_{new}, which are the converted rates for the old and new pages.

Put your answer here.

`2.` Assume under the null hypothesis, p_{new} and p_{old} both have "true" success rates equal to the **converted** success rate regardless of page - that is p_{new} and p_{old} are equal. Furthermore, assume they are equal to the **converted** rate in **ab_data.csv** regardless of the page.

Use a sample size for each page equal to the ones in **ab_data.csv**.

Perform the sampling distribution for the difference in **converted** between the two pages over 10,000 iterations of calculating an estimate from the null.

Use the cells below to provide the necessary parts of this simulation. If this doesn't make complete sense right now, don't worry - you are going to work through the problems below to complete this problem. You can use **Quiz 5** in the classroom to make sure you are on the right track.

a. What is the **convert rate** for p_{new} under the null?

```
#we will use the entire dataset to compute the convertion
#rate of new page because the null hypotheses consider that there is
#no difference between old and new page convertion

p_new=df2['converted'].mean()
p_new
```

```
0.11959708724499628
```

b. What is the **convert rate** for p_{old} under the null?

```
#we will use the entire dataset to compute the convertion
#rate of old page because the null hypotheses consider that there is
#no difference between old and new page convertion
```

```
p_old=df2['converted'].mean()
p_old
```

```
0.11959708724499628
```

c. What is n_{new}?

```
#number of users who landing new page
n_new=len(df2[df2['landing_page']=='new_page'])
n_new
```

```
145310
```

d. What is n_{old}?

```
#number of users who landing old page
n_old=len(df2[df2['landing_page']=='old_page'])
n_old
```

```
145274
```

e. Simulate n_{new} transactions with a convert rate of p_{new} under the null. Store these n_{new} 1's and 0's in **new_page_converted**.

```
new_page_converted = np.random.binomial(1,p_new,n_new)
new_page_converted.mean()
```

```
0.12085885348565137
```

f. Simulate n_{old} transactions with a convert rate of p_{old} under the null. Store these n_{old} 1's and 0's in **old_page_converted**.

```
old_page_converted = np.random.binomial(1,p_old,n_old)
old_page_converted.mean()
```

```
0.1197048336247367
```

g. Find p_{new} - p_{old} for your simulated values from part (e) and (f).

```
new_page_converted.mean()   -   old_page_converted.mean()
```

```
-0.00010535664448757531
```

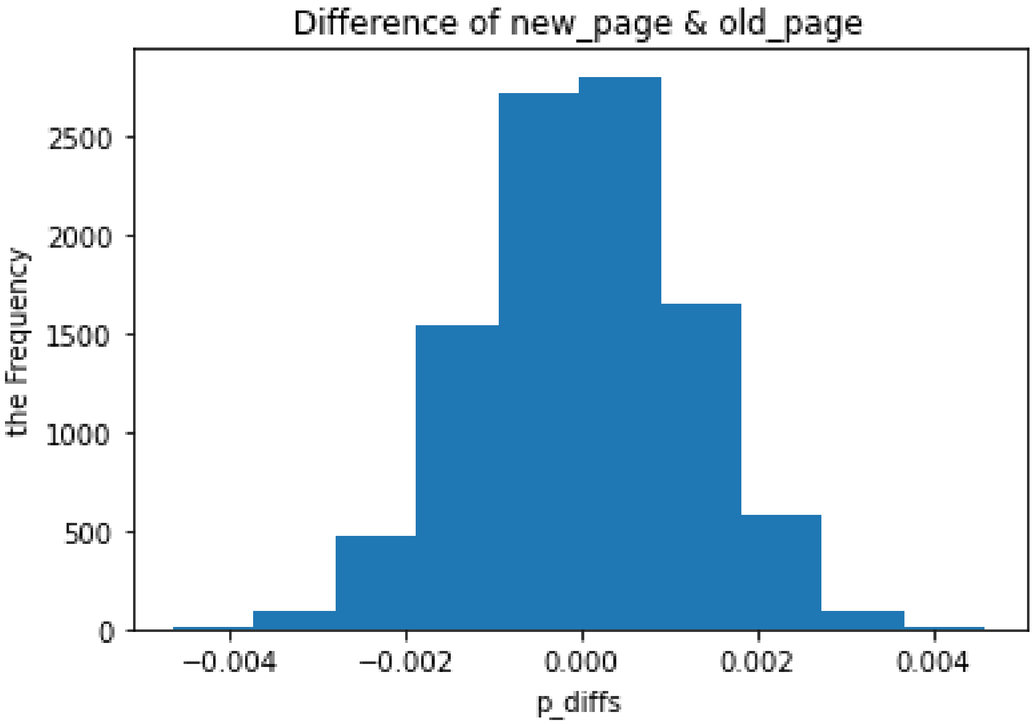
h. Simulate 10,000 p_{new} - p_{old} values using this same process similarly to the one you calculated in parts **a. through g.** above. Store all 10,000 values in a numpy array called **p_diffs**.

```
# Simulate 10,000  pnew  -  pold
p_diffs = []
for _ in range( 10000):

    new_page_converted = np.random.binomial(1,p_new,n_new).mean()
    old_page_converted = np.random.binomial(1,p_old,n_old).mean()
    p_diffs.append(new_page_converted - old_page_converted)
```

i. Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

```
#showing the histogram of the p_diffs.
plt.hist(p_diffs);
plt.xlabel('p_diffs')
plt.ylabel('the Frequency')
plt.title(' Difference of new_page & old_page ');
```



j. What proportion of the **p_diffs** are greater than the actual difference observed in **ab_data.csv**?

```
df_control = df2[df2['group'] == "control"]
df_treatment = df2[df2['group'] == "treatment"]
# display the observed difference in ab_data
observed_diffs = df_treatment.converted.mean() - 
df_control.converted.mean()
observed_diffs
```

```
-0.0015782389853555567
```

```
#proprtion of difference between p_diffs and observed_diffs
(p_diffs > observed_diffs).mean()
```

91% is proportion of the p_diffs are greater than the actual difference observed in ab_data

k. In words, explain what you just computed in part **j.** What is this value called in scientific studies? What does this value mean in terms of whether or not there is a difference between the new and old pages?

this value is called p value and it means that we cannot reject the null hypothesis and we dont have an evidence that the new_page has a higher conversion rate than the old_page.

l. We could also use a built-in to achieve similar results. Though using the built-in might be easier to code, the above portions are a walkthrough of the ideas that are critical to correctly thinking about statistical significance. Fill in the below to calculate the number of conversions for each page, as well as the number of individuals who received each page. Let `n_old` and `n_new` refer the the number of rows associated with the old page and new pages, respectively.

```
import statsmodels.api as sm

n_new = len(df2[df2['landing_page' ]== "new_page"])
n_old = len(df2[df2['landing_page'] == "old_page"])
convert_new = len(df2.query('landing_page == "new_page" & converted == 
1'))
convert_old = len(df2.query('landing_page == "old_page" & converted == 
1'))
print(' convert_new is :',convert_new,'\n',
      'convert_old is :', convert_old,'\n',
      'n_new is :',n_new,'\n',
      'n_old is :', n_old)
```

```
 convert_new is : 17264
 convert_old is : 17489
 n_new is : 145310
 n_old is : 145274
```

m. Now use `stats.proportions_ztest` to compute your test statistic and p-value. Here is a helpful link on using the built in.

```
#computing our statistics with built_in function (statsmodels)
z_score, p_value = sm.stats.proportions_ztest([convert_old, 
convert_new], [n_old, n_new],value=None, alternative='smaller', 
prop_var=False)

print(' z score is : ' ,z_score,'\n','p value is : ', p_value)
```

```
 z score is :  1.3109241984234394
 p value is :  0.9050583127590245
```

n. What do the z-score and p-value you computed in the previous question mean for the conversion rates of the old and new pages? Do they agree with the findings in parts **j.** and **k.**?

> z-score and p_value mean that one doesn't reject the Null.
>
> The Null hypothesis mean the converted rate of the old_page is equal to or greater than the converted rate of the new_page.
>
> The p_value is 0.91 and is higher than 0.05 significance level which is premitted. That means we can not be confident with a 95% confidence level that the converted rate of the new_page is larger than the old_page.
>
> yes its agree (p value is 91% in all result)

Part III - A regression approach

`1.` In this final part, you will see that the result you acheived in the previous A/B test can also be acheived by performing regression.

a. Since each row is either a conversion or no conversion, what type of regression should you be performing in this case?

> it should be logistic regression because the dependant variable is a binary variable

b. The goal is to use **statsmodels** to fit the regression model you specified in part **a.** to see if there is a significant difference in conversion based on which page a customer receives. However, you first need to create a column for the intercept, and create a dummy variable column for which page each user received. Add an **intercept** column, as well as an **ab_page** column, which is 1 when an individual receives the **treatment** and 0 if **control**.

```
df2[['c','treatment']]= pd.get_dummies(df2['group'])

df2 = df2.drop('c',axis = 1)
df2.head()
```

```
   user_id                   timestamp      group landing_page
converted  \
0   851104  2017-01-21 22:11:48.556739    control     old_page
0
1   804228  2017-01-12 08:01:45.159739    control     old_page
0
2   661590  2017-01-11 16:55:06.154213  treatment     new_page
0
3   853541  2017-01-08 18:28:03.143765  treatment     new_page
0
4   864975  2017-01-21 01:52:26.210827    control     old_page
1
```

```
   treatment
0          0
1          0
2          1
3          1
4          0
```

```
df2.rename(columns={'treatment':'ab_page'},inplace=True)
df2
```

```
        user_id                   timestamp      group landing_page  \
0        851104  2017-01-21 22:11:48.556739    control     old_page   
1        804228  2017-01-12 08:01:45.159739    control     old_page   
2        661590  2017-01-11 16:55:06.154213  treatment     new_page   
3        853541  2017-01-08 18:28:03.143765  treatment     new_page   
4        864975  2017-01-21 01:52:26.210827    control     old_page   
...         ...                         ...        ...          ...   
294473   751197  2017-01-03 22:28:38.630509    control     old_page   
294474   945152  2017-01-12 00:51:57.078372    control     old_page   
294475   734608  2017-01-22 11:45:03.439544    control     old_page   
294476   697314  2017-01-15 01:20:28.957438    control     old_page   
294477   715931  2017-01-16 12:40:24.467417  treatment     new_page   

        converted  ab_page  
0               0        0  
1               0        0  
2               0        1  
3               0        1  
4               1        0  
...           ...      ...  
294473          0        0  
294474          0        0  
294475          0        0  
294476          0        0  
294477          0        1  

[290584 rows x 6 columns]
```

c. Use **statsmodels** to import your regression model. Instantiate the model, and fit the model using the two columns you created in part **b.** to predict whether or not an individual converts.

```
from scipy import stats
stats.chisqprob = lambda chisq, df2: stats.chi2.sf(chisq, df2)

df2['intercept'] = 1

lm = sm.Logit(df2['converted'],df2[['intercept','ab_page']])
results = lm.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.366118
         Iterations 6
```

d. Provide the summary of your model below, and use it as necessary to answer the following questions.

```
results.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results                           
                                                                              
======================================================================
========
Dep. Variable:             converted   No. Observations:              
290584
Model:                         Logit   Df Residuals:                  
290582
Method:                          MLE   Df Model:                      
1
Date:               Fri, 09 Sep 2022   Pseudo R-squ.:                 
8.077e-06
Time:                       18:04:51   Log-Likelihood:            -
1.0639e+05
converged:                      True   LL-Null:                   -
1.0639e+05
Covariance Type:           nonrobust   LLR p-value:                   
0.1899
======================================================================
========
                 coef    std err          z      P>|z|      [0.025      
0.975]
----------------------------------------------------------------------
--------
intercept     -1.9888      0.008   -246.669      0.000      -2.005      
-1.973
ab_page       -0.0150      0.011     -1.311      0.190      -0.037      
0.007
======================================================================
========
"""
```

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in **Part II**? **Hint**: What are the null and alternative hypotheses associated with your regression model, and how do they compare to the null and alternative hypotheses in the **Part II**?

p_valueu is 0.190.

> Part II, the p-value was calculated where the null hypothesis was that the convertion rate for the new page more than the old page, and the alternative was the convertion rate for the old page less than or equal to the old page.
>
> Part III, we used variables, and used a linear model to determine the p-value. The null hypothesis was that the difference between the pages = 0, and the alternative hypothesis was the difference between the pages != 0.

f. Now, you are considering other things that might influence whether or not an individual converts. Discuss why it is a good idea to consider other factors to add into your regression model. Are there any disadvantages to adding additional terms into your regression model?

> well it is a good idea to add other factor to the model but it may be get more coplexity and this is consider a disadvantage

g. Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives. You will need to read in the **countries.csv** dataset and merge together your datasets on the approporiate rows. Here are the docs for joining tables.

Does it appear that country had an impact on conversion? Don't forget to create dummy variables for these country columns - **Hint: You will need two columns for the three dummy variables.** Provide the statistical output as well as a written response to answer this question.

```
countries_df = pd.read_csv('./countries.csv')
df_new =
countries_df.set_index('user_id').join(df2.set_index('user_id'),
how='inner')
```

```
#show top rows from countries data
countries_df .head(10)
```

```
   user_id country
0   834778      UK
1   928468      US
2   822059      UK
3   711597      UK
4   710616      UK
5   909908      UK
6   811617      US
7   938122      US
8   887018      US
9   820683      US
```

```
#show data in country column
countries_df['country'].unique()
```

```
array(['UK', 'US', 'CA'], dtype=object)
```

```
#show data after joining two files
df_new.head(10)
```

```
        country                   timestamp      group landing_page  \
user_id
834778       UK  2017-01-14 23:08:43.304998    control     old_page
928468       US  2017-01-23 14:44:16.387854  treatment     new_page
822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
711597       UK  2017-01-22 03:14:24.763511    control     old_page
710616       UK  2017-01-16 13:14:44.000513  treatment     new_page
909908       UK  2017-01-06 20:44:26.334764  treatment     new_page
811617       US  2017-01-02 18:42:11.851370  treatment     new_page
938122       US  2017-01-10 09:32:08.222716  treatment     new_page
887018       US  2017-01-06 11:09:40.487196  treatment     new_page
820683       US  2017-01-14 11:52:06.521342  treatment     new_page

         converted  ab_page  intercept
user_id
834778           0        0          1
928468           0        1          1
822059           1        1          1
711597           0        0          1
710616           0        1          1
909908           0        1          1
811617           1        1          1
938122           1        1          1
887018           0        1          1
820683           0        1          1
```

```
### Create the necessary dummy variables
df_new[['UK','US','CA']]=pd.get_dummies(df_new['country'])
df_new.head()
```

```
        country                   timestamp      group landing_page  \
user_id
834778       UK  2017-01-14 23:08:43.304998    control     old_page
928468       US  2017-01-23 14:44:16.387854  treatment     new_page
822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
711597       UK  2017-01-22 03:14:24.763511    control     old_page
710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

         converted  ab_page  intercept  UK  US  CA
user_id
834778           0        0          1   0   1   0
928468           0        1          1   0   0   1
822059           1        1          1   0   1   0
711597           0        0          1   0   1   0
710616           0        1          1   0   1   0
```

```
### Fit Your Linear Model And Obtain the Results
lm =
sm.Logit(df_new['converted'],df_new[['intercept','ab_page','CA','US']]
)
results = lm.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366113
         Iterations 6

<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results

======================================================================
========
Dep. Variable:              converted   No. Observations:
290584
Model:                          Logit   Df Residuals:
290580
Method:                           MLE   Df Model:
3
Date:                Fri, 09 Sep 2022   Pseudo R-squ.:
2.323e-05
Time:                        18:05:26   Log-Likelihood:            -
1.0639e+05
converged:                       True   LL-Null:                   -
1.0639e+05
Covariance Type:            nonrobust   LLR p-value:
0.1760
======================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
--------
intercept     -2.0300      0.027    -76.249      0.000      -2.082
-1.978
ab_page       -0.0149      0.011     -1.307      0.191      -0.037
0.007
CA             0.0408      0.027      1.516      0.130      -0.012
0.093
US             0.0506      0.028      1.784      0.074      -0.005
0.106
======================================================================
========
"""
```

The p-value for ab_page is 0.191

The null hypothesis was that the difference in means would be 0, and the alternative was the difference would be greater or less than 0. The p-value is still large.so that We fail to reject the null hypothesis.

then the difference in means would be 0

h. Though you have now looked at the individual factors of country and page on conversion, we would now like to look at an interaction between page and country to see if there significant effects on conversion. Create the necessary additional columns, and fit the new model.

Provide the summary results, and your conclusions based on the results.

```
#create interaction columns

df_new['us_ab_page'] = df_new.US *df_new.ab_page
df_new['ca_ab_page'] = df_new.CA *df_new.ab_page



lm =
sm.Logit(df_new['converted'],df_new[['intercept','ab_page','US','us_ab
_page','CA','ca_ab_page']])
results = lm.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 6

<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results

======================================================================
========
Dep. Variable:            converted   No. Observations:
290584
Model:                        Logit   Df Residuals:
290578
Method:                         MLE   Df Model:
5
Date:              Fri, 09 Sep 2022   Pseudo R-squ.:
3.482e-05
Time:                      18:05:54   Log-Likelihood:            -
1.0639e+05
converged:                     True   LL-Null:                   -
1.0639e+05
Covariance Type:          nonrobust   LLR p-value:
0.1920
======================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
--------
intercept     -2.0040      0.036    -55.008      0.000      -2.075
-1.933
ab_page       -0.0674      0.052     -1.297      0.195      -0.169
0.034
US             0.0118      0.040      0.296      0.767      -0.066
0.090
us_ab_page     0.0783      0.057      1.378      0.168      -0.033
0.190
CA             0.0175      0.038      0.465      0.642      -0.056
0.091
ca_ab_page     0.0469      0.054      0.872      0.383      -0.059
0.152
======================================================================
========
"""
```

Summary and conclusion on regression

p_value for both interactions > 0.05.

there is no differecne brtween influence of landing_page in US and other countries.

Conclusions

there is not enough evidence that the new_page increases the conversion rate as compared to the old_page.

no evidence that the countries [US, UK ,CA] influencec the convertion rate.