Analyze A/B Test Results (V1)

This project will assure you have mastered the subjects covered in the statistics lessons. We have organized the current notebook into the following sections:

Specific programming tasks are marked with a **ToDo** tag.

Introduction

A/B tests are very commonly performed by data analysts and data scientists. For this project, you will be working to understand the results of an A/B test run by an e-commerce website. Your goal is to work through this notebook to help the company understand if they should:

- Implement the new webpage,
- Keep the old webpage, or
- Perhaps run the experiment longer to make their decision.

Each **ToDo** task below has an associated quiz present in the classroom. Though the classroom quizzes are **not necessary** to complete the project, they help ensure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the rubric specification.

Part I - Probability

To get started, let's import our libraries.

In [1]:

```
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)
```

ToDo 1.1

Now, read in the `ab_data.csv` data. Store it in `df`. Below is the description of the data, there are a total of 5 columns:

Data columns	Purpose	Valid values
user_id	Unique ID	Int64 values
timestamp	Time stamp when the user visited the webpage	-
group	In the current A/B experiment, the users are categorized into two broad groups. The `control` group users are expected to be served with `old_page` ; and `treatment` group users are matched with the `new_page` . However, **some inaccurate rows** are present in the initial data, such as a `control` group user is matched with a `new_page` .	`['control', 'treatment']`
landing_page	It denotes whether the user visited the old or new webpage.	`['old_page', 'new_page']`
converted	It denotes whether the user decided to pay for the company's product. Here, `1` means yes, the user bought the product.	`[0, 1]`

</center> Use your dataframe to answer the questions in Quiz 1 of the classroom.

---

a. Read in the dataset from the `ab_data.csv` file and take a look at the top few rows here:

In [2]:
```
df = pd.read_csv('ab_data.csv')
df.head()
```

Out[2]:

	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

---

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

In [3]:
```
df.shape
```

Out[3]:
```
(294478, 5)
```

---

c. The number of unique users in the dataset.

In [4]:
```
df.user_id.nunique()
```

Out[4]:
```
290584
```

---

d. The proportion of users converted.

In [5]:
```
df.groupby(['converted']).user_id.count() / df["user_id"].count()
```

Out[5]:
```
converted
0    0.880341
1    0.119659
Name: user_id, dtype: float64
```

---

e. The number of times when the "group" is `treatment` but "landing_page" is not a `new_page` .

In [6]:
```
# V1
df.query("(group == 'treatment' and landing_page == 'old_page')").shape[0]
```

Out[6]:
```
1965
```

---

f. Do any of the rows have missing values?

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   user_id       294478 non-null  int64 
 1   timestamp     294478 non-null  object
 2   group         294478 non-null  object
 3   landing_page  294478 non-null  object
 4   converted     294478 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
```

ToDo 1.2

In a particular row, the **group** and **landing_page** columns should have either of the following acceptable values:

user_id	timestamp	group	landing_page	converted
XXXX	XXXX	`control`	`old_page`	X
XXXX	XXXX	`treatment`	`new_page`	X

It means, the `control` group users should match with `old_page` ; and `treatment` group users should matched with the `new_page` .

However, for the rows where `treatment` does not match with `new_page` or `control` does not match with `old_page` , we cannot be sure if such rows truly received the new or old wepage.

Use **Quiz 2** in the classroom to figure out how should we handle the rows where the group and landing_page columns don't match?

---

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**.

In [8]:
```
# Remove the inaccurate rows, and store the result in a new dataframe df2
df2 = df.drop((df.query("group == 'treatment' and landing_page != 'new_page'").index),
df2 = df2.drop((df2.query("group == 'control' and landing_page != 'old_page'").index),
```

In [9]:
```
# Double Check all of the incorrect rows were removed from df2 -
# Output of the statement below should be 0
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sh
```

Out[9]:
```
0
```

ToDo 1.3

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

---

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

In [10]:
```
df2.user_id.nunique()
```

Out[10]:
```
290584
```

---

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

In [11]:
```
df2.user_id.duplicated().sum()
```

Out[11]:
```
1
```

In [12]:
```
df2[df2.duplicated(['user_id'])]
```

Out[12]:

	user_id	timestamp	group	landing_page	converted
2893	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

---

c. Display the rows for the duplicate **user_id**?

In [13]:
```
df2.query("user_id == 773192")
```

Out[13]:

	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**, from the **df2** dataframe.

In [14]:
```
# Remove one of the rows with a duplicate user_id..
df2.drop_duplicates(subset='user_id', inplace=True)

# Check again if the row with a duplicate user_id is deleted or not
df2.user_id.duplicated().sum()
```

Out[14]:
```
0
```

In [15]:
```
df2.query("user_id == 773192")
```

Out[15]:

	user_id	timestamp	group	landing_page	converted
1899	773192	2017-01-09 05:37:58.781806	treatment	new_page	0

ToDo 1.4

Use **df2** in the cells below 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?

In [16]:
```
n_new = df2.query("group == 'treatment'").user_id.count()
n_old = df2.query("group == 'control'").user_id.count()
n_all = df2.user_id.count()
n_new , n_old, n_all
```

Out[16]:
```
(145310, 145274, 290584)
```

In [17]:
```
p_pop = df2.query("converted == 1").user_id.count() / n_all
p_pop
```

Out[17]:
```
0.11959708724499628
```

---

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

In [18]:
```
p_ctr = df2.query("converted == 1 and group == 'control'").user_id.count() / n_old
p_ctr
```

Out[18]:
```
0.1203863045004612
```

---

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

In [19]:
```
p_trt = df2.query("converted == 1 and group == 'treatment'").user_id.count() / n_new
p_trt
```

Out[19]:
```
0.11880806551510564
```

In [20]:
```
# Calculate the actual difference (obs_diff) between the conversion rates for the two g
obs_diff = p_trt - p_ctr
obs_diff
```

Out[20]:
```
-0.0015782389853555567
```

---

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

In [21]:
```
df2.query("group == 'treatment'").user_id.count() / n_all
```

Out[21]:
```
0.5000619442226688
```

---

e. Consider your results from parts (a) through (d) above, and explain below whether the new `treatment` group users lead to more conversions.

> **Conclusions so far:**
>
> It makes sense to keep the old page since the conversion rate for the treatment group is slightly lower than the rate for the control group.

Part II - A/B Test

Since a timestamp is associated with each event, you could run a hypothesis test continuously as long as you observe the events.

However, then the hard questions would be:

- 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.

ToDo 2.1

For now, consider you need to make the decision just based on all the data provided.

> Recall that you just calculated that the "converted" probability (or rate) for the old page is *slightly* higher than that of the new page (ToDo 1.4.c).

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 be your null and alternative hypotheses (H_0 and H_1)?

You can state your hypothesis in terms of words or in terms of p_{old} and p_{new}, which are the "converted" probability (or rate) for the old and new pages respectively.

> **Our hypothesis Test:**
>
> - H0 : P_old - P_new >= 0
> - H1 : P_old - P_new < 0

ToDo 2.2 - Null Hypothesis H_0 Testing

Under the null hypothesis H_0, assume that p_{new} and p_{old} are equal. Furthermore, assume that p_{new} and p_{old} both are equal to the **converted** success rate in the `df2` data regardless of the page. So, our assumption is:

$$p_{new} = p_{old} = p_{population}$$

In this section, you will:

- Simulate (bootstrap) sample data set for both groups, and compute the "converted" probability p for those samples.
- Use a sample size for each group equal to the ones in the `df2` data.
- Compute the difference in the "converted" probability for the two samples above.
- Perform the sampling distribution for the "difference in the converted probability" between the two simulated-samples over 10,000 iterations; and calculate an estimate.

Use the cells below to provide the necessary parts of this simulation. You can use **Quiz 5** in the classroom to make sure you are on the right track.

---

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

In [22]:

```
p_new = df2.query("converted == 1").user_id.count() / n_all
p_new
```

Out[22]:

```
0.11959708724499628
```

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

In [23]:
```
p_old = df2.query("converted == 1").user_id.count() / n_all
p_old
```

Out[23]:
```
0.11959708724499628
```

c. What is n_{new}, the number of individuals in the treatment group?

In [24]:
```
df2.query("group == 'treatment'").user_id.count() # n_new
```

Out[24]:
```
145310
```

d. What is n_{old}, the number of individuals in the control group?

In [25]:
```
df2.query("group == 'control'").user_id.count() # n_old
```

Out[25]:
```
145274
```

e. Simulate Sample for the `treatment` Group
Simulate n_{new} transactions with a conversion rate of p_{new} under the null hypothesis.

In [26]:
```
# Simulate a Sample for the treatment Group

new_page_converted = np.random.choice([1, 0], size=n_new, p = [p_new, 1-p_new])

Pss_new = np.mean(new_page_converted)
Pss_new
```

Out[26]:
```
0.11829192760305554
```

f. Simulate Sample for the `control` Group
Simulate n_{old} transactions with a conversion rate of p_{old} under the null hypothesis.
Store these n_{old} 1's and 0's in the `old_page_converted` numpy array.

In [27]:
```
# Simulate a Sample for the control Group

old_page_converted = np.random.choice([1, 0], size=n_old, p = [p_old, 1-p_old])

Pss_old = np.mean(old_page_converted)
Pss_old
```

Out[27]:

```
0.12051709184024671
```

---

g. Find the difference in the "converted" probability $(p'_{new}$ - $p'_{old})$ for your simulated samples from the parts (e) and (f) above.

In [28]:

```
obs_diff = Pss_new - Pss_old
obs_diff
```

Out[28]:

```
-0.002225164237191171
```

---

h. Sampling distribution

Re-create `new_page_converted` and `old_page_converted` and find the $(p'_{new}$ - $p'_{old})$ value 10,000 times using the same simulation process you used in parts (a) through (g) above.

Store all $(p'_{new}$ - $p'_{old})$ values in a NumPy array called `p_diffs` .

In [29]:

```
# Sampling distribution
p_diffs = []

for _ in range(10000):
    pss1_new = np.random.choice([1, 0],n_new,replace = True,p = [p_new, 1-p_new])
    pss1_old = np.random.choice([1, 0],n_old,replace = True,p = [p_old, 1-p_old])
    pss2_new = pss1_new.mean()
    pss2_old = pss1_old.mean()
    p_diffs.append(pss2_new-pss2_old)
```

---

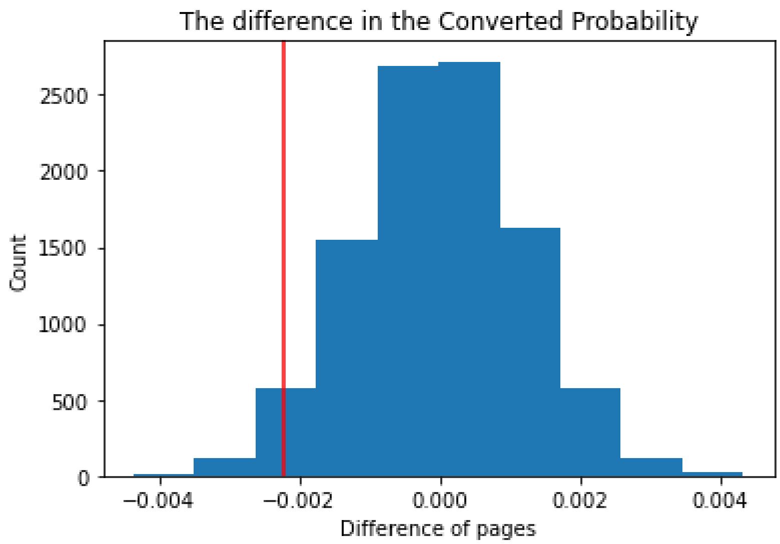
i. Histogram

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.

Also, use `plt.axvline()` method to mark the actual difference observed in the `df2` data (recall `obs_diff`), in the chart.

In [30]:

```
plt.hist(p_diffs)
plt.title('The difference in the Converted Probability')
plt.xlabel('Difference of pages')
plt.ylabel('Count')
plt.axvline(x= obs_diff, color='red');
```



---

j. What proportion of the **p_diffs** are greater than the actual difference observed in the `df2` data?

In [31]:

```
p_diff = p_trt - p_ctr
counter = 0
for i in p_diffs:
    if i> p_diff:
        counter = counter + 1

p_value = counter / (len(p_diffs))
p_value
```

Out[31]:

```
0.9066
```

---

k. Please explain in words what you have just computed in part **j** above.

- What is this value called in scientific studies?
- What does this value signify in terms of whether or not there is a difference between the new and old pages? *Hint*: Compare the value above with the "Type I error rate (0.05)".

> **Conclusions so far:**
>
> - This value represents the p-value.
> - As the p-value of more than 90% is over the significant level 5%, we failed to reject the null hypothesis. So we better keep the old page.

---

l. Using Built-in Methods for Hypothesis Testing
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 statements below to calculate the:

- `convert_old` : number of conversions with the old_page
- `convert_new` : number of conversions with the new_page
- `n_old` : number of individuals who were shown the old_page
- `n_new` : number of individuals who were shown the new_page

In [32]:
```
import statsmodels.api as sm

# number of conversions with the old_page
convert_old = df2.query("converted == 1 and group == 'control'").user_id.count()

# number of conversions with the new_page
convert_new = df2.query("converted == 1 and group == 'treatment'").user_id.count()

# number of individuals who were shown the old_page
n_old = df2.query("group == 'control'").user_id.count()

# number of individuals who received new_page
n_new = df2.query("group == 'treatment'").user_id.count()
```

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

The syntax is:

```
proportions_ztest(count_array, nobs_array, alternative='larger')
```

where,

- `count_array` = represents the number of "converted" for each group
- `nobs_array` = represents the total number of observations (rows) in each group
- `alternative` = choose one of the values from `[‘two-sided’, ‘smaller’, ‘larger’]` depending upon two-tailed, left-tailed, or right-tailed respectively.

The built-in function above will return the z_score, p_value.

About the two-sample z-test

Recall that you have plotted a distribution `p_diffs` representing the difference in the "converted" probability $(p'_{new} - p'_{old})$ for your two simulated samples 10,000 times.

Another way for comparing the mean of two independent and normal distribution is a **two-sample z-test**. You can perform the Z-test to calculate the Z_score, as shown in the equation below:

$$Z_{score} = \frac{(p'_{new} - p'_{old}) - (p_{new} - p_{old})}{\sqrt{\frac{\sigma^2_{new}}{n_{new}} + \frac{\sigma^2_{old}}{n_{old}}}}$$

where,

- p' is the "converted" success rate in the sample

- p_{new} and p_{old} are the "converted" success rate for the two groups in the population.
- σ_{new} and σ_{new} are the standard deviation for the two groups in the population.
- n_{new} and n_{old} represent the size of the two groups or samples (it's same in our case)

> Z-test is performed when the sample size is large, and the population variance is known. The z-score represents the distance between the two "converted" success rates in terms of the standard error.

Next step is to make a decision to reject or fail to reject the null hypothesis based on comparing these two values:

- Z_{score}
- Z_{α} or $Z_{0.05}$, also known as critical value at 95% confidence interval. $Z_{0.05}$ is 1.645 for one-tailed tests, and 1.960 for two-tailed test. You can determine the Z_{α} from the z-table manually.

Decide if your hypothesis is either a two-tailed, left-tailed, or right-tailed test. Accordingly, reject OR fail to reject the null based on the comparison between Z_{score} and Z_{α}.

In other words, we determine whether or not the Z_{score} lies in the "rejection region" in the distribution. A "rejection region" is an interval where the null hypothesis is rejected if the Z_{score} lies in that region.

Reference:

- Example 9.1.2 on this page/09%3A_Two-Sample_Problems/9.01%3A_Comparison_of_Two_Population_Means-_Large_Independent_Samples), courtesy www.stats.libretexts.org

---

In [33]:

```
import statsmodels.api as sm
# ToDo: Complete the sm.stats.proportions_ztest() method arguments
 # V1
z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_old
print("z_score=", z_score)
print("p_value=", p_value)
```

```
z_score= -1.3109241984234394
p_value= 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.**?

> **Conclusions so far:**
>
> - Our previous calculation of p_value yielded the same results.
> - We are still unable to reject the null hypothesis.

Part III - A regression approach

ToDo 3.1

In this final part, you will see that the result you achieved in the A/B test in Part II above can also be achieved by performing regression.

a. Since each row in the `df2` data is either a conversion or no conversion, what type of regression should you be performing in this case?

> **Conclusions so far:**
>
> - Due to the yes/no nature of the problem, we will use logistic regression.

b. The goal is to use **statsmodels** library to fit the regression model you specified in part **a.** above to see if there is a significant difference in conversion based on the page-type a customer receives. However, you first need to create the following two columns in the `df2` dataframe:

1. `intercept` - It should be `1` in the entire column.
2. `ab_page` - It's a dummy variable column, having a value `1` when an individual receives the **treatment**, otherwise `0`.

In [34]:

```
dfr = df2
dfr['intercept'] = 1
dfr['ab_page'] = 0

# Change ab_page cells to 1 when group is treatment
dfr.loc[(dfr['group'] == "treatment"),'ab_page'] = 1

dfr.head(10)
```

Out[34]:

	user_id	timestamp	group	landing_page	converted	intercept	ab_page
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	0
5	936923	2017-01-10 15:20:49.083499	control	old_page	0	1	0
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1	1	1
7	719014	2017-01-17 01:48:29.539573	control	old_page	0	1	0
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1	1	1
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1	1	1

c. Use **statsmodels** to instantiate your regression model on the two columns you created in part (b). above, then fit the model to predict whether or not an individual converts.

In [35]:
```
# import statsmodels.api as sm
smodel = sm.Logit(dfr['converted'], dfr[['intercept','ab_page']])
results = smodel.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.

In [36]:
```
results.summary()
```

Out[36]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290582
Method:	MLE	**Df Model:**	1
Date:	Thu, 16 Jun 2022	**Pseudo R-squ.:**	8.077e-06
Time:	00:06:04	**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**?

> **Conclusions so far:**
>
> - The regression model calculates a p-value of 0.1899 which is lower than that obtained using the z-test.
> - This p-value is probably closer to the true one.

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?

> **Conclusions so far:**
>
> - Different factors can influence the final results of our study, and granting more factors could result in better results.
> - Additional irrelevant factors may result in a destracted result and will waste time.

---

g. Adding countries
Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives in.

1. You will need to read in the **countries.csv** dataset and merge together your `df2` datasets on the appropriate rows. You call the resulting dataframe `df_merged`. Here are the docs for joining tables.

2. Does it appear that country had an impact on conversion? To answer this question, consider the three unique values, `['UK', 'US', 'CA']`, in the `country` column. Create dummy variables for these country columns.

Provide the statistical output as well as a written response to answer this question.

In [37]:

```
# Read the countries.csv
countries = pd.read_csv('countries.csv')
countries.head()
```

Out[37]:

	user_id	country
0	834778	UK
1	928468	US
2	822059	UK
3	711597	UK
4	710616	UK

In [38]:

```
countries.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 290584 entries, 0 to 290583
Data columns (total 2 columns):
 #   Column    Non-Null Count   Dtype 
---  ------    --------------   ----- 
 0   user_id   290584 non-null  int64 
 1   country   290584 non-null  object
```

```
dtypes: int64(1), object(1)
memory usage: 4.4+ MB
```

In [39]:
```
dfr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290584 entries, 0 to 294477
Data columns (total 7 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   user_id       290584 non-null  int64
 1   timestamp     290584 non-null  object
 2   group         290584 non-null  object
 3   landing_page  290584 non-null  object
 4   converted     290584 non-null  int64
 5   intercept     290584 non-null  int64
 6   ab_page       290584 non-null  int64
dtypes: int64(4), object(3)
memory usage: 17.7+ MB
```

In [40]:
```
# Join with the dfr dataframe
df_merged = dfr.merge(countries, on='user_id', how='left')
df_merged.head()
```

Out[40]:

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

In [41]:
```
# Create the necessary dummy variables
df_merged['US'] = 0
df_merged['UK'] = 0
df_merged['CA'] = 0

# Change country columns data
df_merged.loc[(df_merged['country'] == 'US'), 'US'] = 1
df_merged.loc[(df_merged['country'] == 'UK'), 'UK'] = 1
df_merged.loc[(df_merged['country'] == 'CA'), 'CA'] = 1

df_merged.head(10)
```

Out[41]:

	user_id	timestamp	group	landing_page	converted	intercept	ab_page	country	US	UK
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	0	US	1	0

	user_id	timestamp	group	landing_page	converted	intercept	ab_page	country	US	UK
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	0	US	1	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1	US	1	0
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1	US	1	0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	0	US	1	0
5	936923	2017-01-10 15:20:49.083499	control	old_page	0	1	0	US	1	0
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1	1	1	CA	0	0
7	719014	2017-01-17 01:48:29.539573	control	old_page	0	1	0	US	1	0
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1	1	1	UK	0	1
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1	1	1	CA	0	0

In [42]:

```
# V1
# Fit your model, and summarize the results
smodel2 = sm.Logit(df_merged['converted'], df_merged[['intercept', 'US', 'UK']])
results2 = smodel2.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.366116
         Iterations 6
```

In [43]:

```
# V1
results2.summary()
```

Out[43]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290581
Method:	MLE	**Df Model:**	2
Date:	Thu, 16 Jun 2022	**Pseudo R-squ.:**	1.521e-05
Time:	00:06:07	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.1984

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-2.0375	0.026	-78.364	0.000	-2.088	-1.987

US	0.0408	0.027	1.518	0.129	-0.012	0.093
UK	0.0507	0.028	1.786	0.074	-0.005	0.106

> **Conclusions so far:**
>
> - There was some effect of country on conversion rate, but not enough to be statistically significant.

---

h. Fit your model and obtain the results

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 are there significant effects on conversion. **Create the necessary additional columns, and fit the new model.**

Provide the summary results (statistical output), and your conclusions (written response) based on the results.

In [44]:

```
# V1
df_merged['ab_UK'] = df_merged['ab_page'] * df_merged['UK']
df_merged['ab_US'] = df_merged['ab_page'] * df_merged['US']
df_merged.head()
```

Out[44]:

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

In [45]:

```
# V1
# Fit your model, and summarize the results
smodel3 = sm.Logit(df_merged['converted'], df_merged[['intercept', 'ab_page', 'ab_US',
results3 = smodel3.fit()
```

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

In [46]:

```
# V1
results3.summary()
```

Out[46]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290580
Method:	MLE	**Df Model:**	3
Date:	Thu, 16 Jun 2022	**Pseudo R-squ.:**	3.351e-05
Time:	00:06:09	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.06785

	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	-0.0827	0.038	-2.176	0.030	-0.157	-0.008
ab_US	0.0644	0.038	1.679	0.093	-0.011	0.140
ab_UK	0.0901	0.040	2.225	0.026	0.011	0.169

> **Conclusions so far:**
>
> - The interaction between page and country has good effect on conversion rate, but it is still not enough to be statistically significant.

Final Conclusion

> - Throughout all of our tests here, we never got a p_value lower than 0.05, including those of the countries (US and UK), which means we were unable to reject the null hypothesis.
> - Therefore, we can conclude that the new version of the page is not better than the old version.
> - It is possible that we will get better results if we continue to test for longer period of time.
> - In order for the new page to be more attractive and capture the attention of more clients, it needs more enhancements.

In [ ]: