Analyze_ab_test_results_notebook

February 21, 2022

1 Analyze A/B Test

2 Analyzed By Odai Alsalieti

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

- Introduction
- Part I - Probability
- Part II - A/B Test
- Part III - Regression
- Final Check
- Conclusion
- Sources

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.

Part I - Probability

To get started, let's import our libraries.

[1]:
```
# 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)
```

2.0.1 ToDo 1.1

Now, read in the `ab_data.csv` data. Store it in `df`.

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

[2]:
```
# Read the dataset
df = pd.read_csv('ab_data.csv')
```

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

[3]:
```
# Find the number of rows
df.shape[0]
```

[3]:
```
294478
```

c. The number of unique users in the dataset.

[4]:
```
# Number of unique users
df.user_id.nunique()
```

[4]:
```
290584
```

d. The proportion of users converted.

[5]:
```
# The proportion of users converted.
df.converted.mean()
```

[5]:
```
0.11965919355605512
```

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

[6]:
```
#The number of times when:
# the "group" is treatment but "landing_page" is not a new_page
df.query('group == "treatment" and landing_page != "new_page"').shape[0]
```

[6]:
```
1965
```

f. Do any of the rows have missing values?

[7]:
```
# Check of missing values
df.isnull().sum()
```

[7]:
```
user_id         0
timestamp       0
group           0
landing_page    0
converted       0
dtype: int64
```

2.0.2 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**.

[8]:
```
# Remove the inaccurate rows, and store the result in a new dataframe df2
df2 = pd.concat(
    [df.query('group == "treatment" and landing_page == "new_page"'),
    df.query('group == "control" and landing_page == "old_page"')]
)
df2.head(3)
```

[8]:

	user_id	timestamp	group	landing_page	converted
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
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1

[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].shape[0]
```

[9]:
```
0
```

2.0.3 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**?

[10]:
```
# Check of unique user_ids
df2.user_id.nunique()
```

[10]:
```
290584
```

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

[11]:
```
# check repeated user_id in df2
df2[df2['user_id'].duplicated()]
```

[11]:

	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**?

[12]:
```
# Display the rows for the duplicate user_id
df2[df2['user_id'].duplicated(keep= False)]
```

[12]:

	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.

[13]:
```
# drop duplicated user_id by index number
df2 = df2.drop(2893)
# Check again if the row with a duplicate user_id is deleted or not
df2[df2['user_id'].duplicated()]
```

[13]:
```
Empty DataFrame
Columns: [user_id, timestamp, group, landing_page, converted]
Index: []
```

2.0.4 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?

$p_{population}$.

[14]:
```
# The proportion of users converted
df2.converted.mean()
```

[14]:
```
0.11959708724499628
```

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

[15]:
```
# probability control group converted
p_control_converted = df2.query('group == "control"')['converted'].mean()
p_control_converted
```

[15]:
```
0.1203863045004612
```

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

[16]:
```
# probability treatement group converted
p_treatment_converted = df2.query('group == "treatment"')['converted'].mean()
p_treatment_converted
```

[16]:
```
0.11880806551510564
```

[17]:
```
# Calculate the actual difference (obs_diff) between the conversion rates for␣
↪the two groups.
obs_diff = p_treatment_converted - p_control_converted
obs_diff
```

[17]:
```
-0.0015782389853555567
```

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

[18]:
```
# the probability that an individual received the new page
df2[df2['group']=='treatment']['user_id'].shape[0] / df2['user_id'].shape[0]
```

[18]:
```
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.

3 Answer

We note through our calculation of the probabilities of: - Treatment : 0.118807247902774058 - The control group: 0.1203863045004612 Although the probability (or average) "converted" of the old page is slightly higher than the new page by 0.0015, we note that the conversion ratio of both groups has no difference between them, we cannot say that the new page leads to more conversions, we need to Searching for other factors

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.

3.0.1 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.

4 Answer

$ H_0 : **p_{new}** - **p_{old}** <= 0 < br > **H_1** : **p_{new}** - **p_{old}$** > 0

4.0.1 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?

[19]:
```
# the conversion rate for      under the null hypothesis
p_new = df2.converted.mean()
p_new
```

[19]: 0.11959708724499628

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

[20]:
```
# the conversion rate for      under the null hypothesis
p_old = df2.converted.mean()
p_old
```

[20]: 0.11959708724499628

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

[21]:
```
# the number of individuals in the treatment group
n_new = df2.query('group == "treatment"').shape[0]
n_new
```

[21]: 145310

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

[22]:
```
# the number of individuals in the control group
n_old = df2.query('group == "control"').shape[0]
n_old
```

[22]: 145274

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

[23]:
```
# Simulate a Sample for the treatment Group
new_page_converted = np.random.choice([0, 1], size=n_new, p=[(1 - p_new),␣
↪p_new])
new_page_converted.mean()
```

[23]: 0.11922097584474571

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.

[24]:
```
# Simulate a Sample for the control Group
old_page_converted = np.random.choice([0, 1], size=n_new, p=[(1 - p_new),␣
↪p_new])
old_page_converted.mean()
```

[24]: 0.11922785768357305

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

[25]:
```
# Find the difference in the "converted" probability
diffs = new_page_converted.mean() - old_page_converted.mean()
diffs
```

[25]: -6.881838827341169e-06

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.

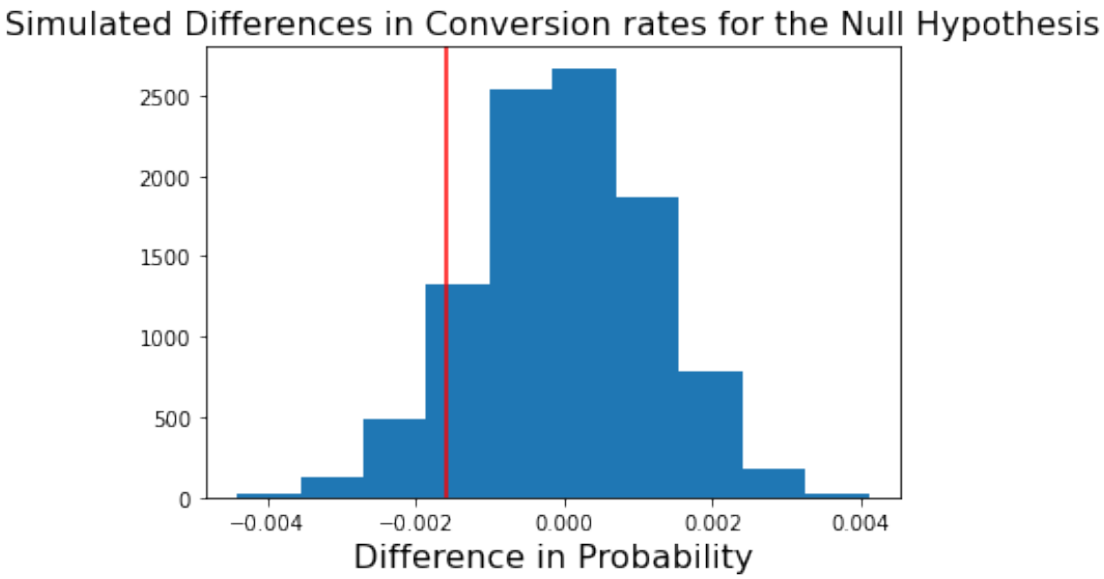
Store all (p'_{new} - p'_{old}) values in a NumPy array called `p_diffs`.

[26]:
```
# Sampling distribution
p_diffs = []

for _ in range(10000):
    new_page_converted = np.random.binomial(n_new, p_new, 10000)/n_new
    old_page_converted = np.random.binomial(n_old, p_old, 10000)/n_old
    p_diffs = new_page_converted - old_page_converted

p_diffs = np.array(p_diffs)
```

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.

[27]:
```
plt.hist(p_diffs)
plt.title("Simulated Differences in Conversion rates for the Null Hypothesis",␣
 ↪fontsize=16)
plt.xlabel("Difference in Probability", fontsize=16)
plt.axvline(obs_diff, color='r');
```



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

[28]:
```
# calculate the p)value
(p_diffs > obs_diff).mean()
```

[28]: 0.9038

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)".

5 Answer

The value computed in the j part is the p_value that is used in hypothesis testing to help us support or reject the null hypothesis. The p_value is the evidence against the null hypothesis. The smaller the p_value, the stronger the evidence that you should reject the null hypothesis.. In our studied case, is the probability of observing the difference between conversion rates between the control and treatment groups, in order to reject the null hypothesis, the p_value must be less than our level of 0.05, but we found it to be 0.9 which is high to allows us to reject the null hypothesis

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 walk-through 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

[29]:
```
import statsmodels.api as sm

# number of conversions with the old_page
convert_old = df2.query('landing_page == "old_page" &␣
↪converted==1')['converted'].shape[0]

# number of conversions with the new_page
convert_new = df2.query('landing_page == "new_page" &␣
↪converted==1')['converted'].shape[0]

# number of individuals who were shown the old_page
n_old = df2.query('group == "control"').shape[0]

# number of individuals who received new_page
n_new = df2.query('group == "treatment"').shape[0]

convert_old, convert_new, n_old, n_new
```

[29]:
```
(17489, 17264, 145274, 145310)
```

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.

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

Reference: - Example 9.1.2 on this page, courtesy www.stats.libretexts.org

[30]:
```
import statsmodels.api as sm
# ToDo: Complete the sm.stats.proportions_ztest() method arguments
z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old],␣
 ↪[n_new, n_old], alternative='larger')
print(z_score, p_value)
```

```
-1.3109241984234394 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.**?

6 Answer

The objective of Z-Test is to check if the observations remain the same or not if the null hypothesis is true it is measured in terms of standard deviations from the mean in our case we have negative Z-score: -1.31092419842 indicating it is below the mean. and we getting a similar p-value to one computed in Hypothesis testing i.e 90%

Part III - A regression approach

6.0.1 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?

7 Answer

In this case We should 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`.

[31]:
```
#  Fit the regression model
# set intercept to 1
df2['intercept'] = 1
# dummy variable
df2[['ab_page', 'old_page']] = pd.get_dummies(df2['landing_page'])
df2.head(2)
```

[31]:
```
   user_id                   timestamp      group landing_page  converted  \
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

   intercept  ab_page  old_page
2          1        1         0
3          1        1         0
```

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.

[32]:
```
log_model = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
results = log_model.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.

[33]:
```
results.summary2()
```

[33]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.000
Dependent Variable: converted        AIC:              212780.3502
Date:               2022-02-21 03:39 BIC:              212801.5095
No. Observations:   290584           Log-Likelihood:   -1.0639e+05
Df Model:           1                LL-Null:          -1.0639e+05
Df Residuals:       290582           LLR p-value:      0.18988
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
-----------------------------------------------------------------
              Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
-----------------------------------------------------------------
intercept    -1.9888    0.0081  -246.6690  0.0000  -2.0046  -1.9730
ab_page      -0.0150    0.0114    -1.3109  0.1899  -0.0374   0.0074
=================================================================

"""
```

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in **Part II**?

8 Answer

The probability value calculated here is 0.189. This is because logarithmic regression is based on a two-tailed test, and the value is still greater than our alpha value of 0.190 $>$ 0.05, so we cannot reject our null hypothesis.

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?

9 Answer

In this case, we only look at the conversion rate from the old page to the new page, but there may be other factors related to the conversion rate, such as users who may not like the changes or are confused by some changes to the new page or perhaps even the geographic location and nationality of each user, is required Adding new factors to improve the model as long as we don't have highly correlated or overlapping predictions

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.

[34]:
```
# Read the countries.csv
df_country = pd.read_csv('countries.csv')
df_country.head()
```

[34]:
```
   user_id country
0   834778      UK
1   928468      US
2   822059      UK
3   711597      UK
4   710616      UK
```

[35]:
```
# Join with the df2 dataframe
df_merged = df_country.set_index('user_id').join(df2.set_index('user_id'),␣
 ↪how='inner')
df_merged.head()
```

[35]:
```
         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  intercept  ab_page  old_page
user_id
834778           0          1        0         1
928468           0          1        1         0
822059           1          1        1         0
711597           0          1        0         1
710616           0          1        1         0
```

[36]:
```
# first we calculate timestamp to check the period of testing
start_time = pd.Timestamp(df2.timestamp.min())
end_time = pd.Timestamp(df2.timestamp.max())
print(end_time - start_time)
```

```
21 days 23:59:49.081927
```

[37]:
```
#check country unique values to set dummies
df_merged.country.unique()
```

[37]:
```
array(['UK', 'US', 'CA'], dtype=object)
```

[38]:
```
# Create the necessary dummy variables
df_merged[['ca', 'uk', 'us']] = pd.get_dummies(df_merged['country'])
df_merged.head()
```

[38]:

user_id	country	timestamp	group	landing_page	converted	intercept	ab_page	old_page	ca	uk	us
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0	1	0	1	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	1	0	0	0	1
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	1	0	0	1	0
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0	1	0	1	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	1	0	0	1	0

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.

[39]:
```
df_merged['US_ab_page'] = df_merged['us'] * df_merged['ab_page']
df_merged['CA_ab_page'] = df_merged['ca'] * df_merged['ab_page']
```

[40]:
```
df_merged.head()
```

[40]:

user_id	country	timestamp	group	landing_page	converted	intercept	ab_page	old_page	ca	uk	us	US_ab_page	\
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									

```
834778           0          1         0         1    0    1    0           0
928468           0          1         1         0    0    0    1           1
822059           1          1         1         0    0    1    0           0
711597           0          1         0         1    0    1    0           0
710616           0          1         1         0    0    1    0           0

         CA_ab_page
user_id
834778            0
928468            0
822059            0
711597            0
710616            0
```

[41]:
```
# Fit your model, and summarize the results
log_model = sm.Logit(df_merged['converted'], df_merged[['intercept', 'uk',␣
↪'us', 'US_ab_page', 'CA_ab_page']])
result = log_model.fit()
result.summary2()
```

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

[41]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.000
Dependent Variable: converted        AIC:              212780.8857
Date:               2022-02-21 03:39 BIC:              212833.7840
No. Observations:   290584           Log-Likelihood:   -1.0639e+05
Df Model:           4                LL-Null:          -1.0639e+05
Df Residuals:       290579           LLR p-value:      0.12653
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
-----------------------------------------------------------------
               Coef.   Std.Err.     z      P>|z|    [0.025   0.975]
-----------------------------------------------------------------
intercept     -2.0040    0.0364  -55.0077  0.0000  -2.0754  -1.9326
uk             0.0172    0.0382    0.4504  0.6524  -0.0576   0.0920
us             0.0175    0.0377    0.4652  0.6418  -0.0563   0.0914
US_ab_page    -0.0206    0.0137   -1.5052  0.1323  -0.0473   0.0062
CA_ab_page    -0.0674    0.0520   -1.2967  0.1947  -0.1694   0.0345
=================================================================

"""
```

10 Answer

By noting the probabilities of all variables, we find that they are greater than 0.05, we conclude that the interactions between the page and the country have no effect on the conversion.

Conclusion

After collecting and analyzing the data, we found no evidence that the new page is better than the old page Although there could be other factors that might change our results, such as giving the model more time to experiment (it was 21 days 23:59:49.081927) or adding other resources As a final result, we failed to reject the null hypothesis so we recommend against switching

sources: - https://pandas.pydata.org/docs/reference/api/pandas.concat.html

- https://www.geeksforgeeks.org/how-to-drop-rows-in-pandas-dataframe-by-index-labels/
- https://stackoverflow.com/questions/43348194/pandas-select-rows-if-id-appear-several-time
- https://stackoverflow.com/questions/22923775/calculate-time-difference-between-two-pandas-columns-in-hours-and-minutes

[42]:
```
from subprocess import call
call(['python', '-m', 'nbconvert' ,'--to', 'html',␣
 ↪'Analyze_ab_test_results_notebook.ipynb',])
call(['python', '-m', 'nbconvert' ,'--to', 'pdf',␣
 ↪'Analyze_ab_test_results_notebook.ipynb',])
```

```
[NbConvertApp] Converting notebook Analyze_ab_test_results_notebook.ipynb to
html
[NbConvertApp] Writing 679500 bytes to Analyze_ab_test_results_notebook.html
[NbConvertApp] Converting notebook Analyze_ab_test_results_notebook.ipynb to pdf
[NbConvertApp] Support files will be in Analyze_ab_test_results_notebook_files/
[NbConvertApp] Making directory ./Analyze_ab_test_results_notebook_files
[NbConvertApp] Writing 83565 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 123802 bytes to Analyze_ab_test_results_notebook.pdf
```

[42]: 0