Analyze A/B Test Results

You may either submit your notebook through the workspace here, or you may work from your local machine and submit through the next page. Either way assure that your code passes the project RUBRIC (https://review.udacity.com/#!/projects/37e27304-ad47-4eb0-a1ab-8c12f60e43d0/rubric). **Please save regularly.**

This project will assure you have mastered the subjects covered in the statistics lessons. The hope is to have this project be as comprehensive of these topics as possible. Good luck!

Table of Contents

Introduction

A/B tests are very commonly performed by data analysts and data scientists. It is important that you get some practice working with the difficulties of these

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 page, keep the old page, or perhaps run the experiment longer to make their decision.

As you work through this notebook, follow along in the classroom and answer the corresponding quiz questions associated with each question. The labels for each classroom concept are provided for each question. This will assure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the criteria. As a final check, assure you meet all the criteria on the RUBRIC (https://review.udacity.com/#!/projects/37e27304-ad47-4eb0-a1ab-8c12f60e43d0/rubric).

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)
```

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:

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]:
```
# shape return number of rows and columns in the dataset. Ex: (294478, 5)
print('Number of rows in the dataset : ', df.shape[0])
```

```
Number of rows in the dataset :  294478
```

c. The number of unique users in the dataset.

In [4]:
```
# counts the numbe rof unique userd by user_id
print('The number of unique users: ', df['user_id'].nunique())
df.shape
```

```
The number of unique users:  290584
```

Out[4]:
```
(294478, 5)
```

d. The proportion of users converted.

In [5]:
```
all_convs = df.query('converted == 1') # selecting all rows that is converted
prop_convs =  100 * all_convs['user_id'].count() / df.shape[0] # calculating proportion
print('The proportion of users converted: ', str(round(prop_convs)) + '%') # rounding the value to int a
```

```
The proportion of users converted:  12%
```

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

In [6]:
```
# df['landing_page'].unique()
a = df[(df['group'] == 'treatment') & (df['landing_page'] == 'old_page')].count()[0]
b = df[(df['group'] == 'control') & (df['landing_page'] == 'new_page')].count()[0]
c = df[(df['group'] == 'treatment') & (df['landing_page'] == 'new_page')].count()[0]
d = df[(df['group'] == 'control') & (df['landing_page'] == 'old_page')].count()[0]
print('The numbe rof times the new_page and treatment dont match: ', a+b) # 1st option - straight
print(df.shape[0] - c - d) # 2nd option - reverse
```

```
The numbe rof times the new_page and treatment dont match:  3893
3893
```

f. Do any of the rows have missing values?

In [7]:
```
# 1st option
df.isnull().sum(axis=1) > 0

# simple one
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
```

NULL/Missing values isn't found in the above dataset

`2.` For the rows where **treatment** does not match with **new_page** or **control** does not match with **old_page**, we cannot be sure if this row truly received the new or old page. Use **Quiz 2** in the classroom to figure out 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.**

In [8]:
```
df2 = df[(df['group'] == 'treatment') & (df['landing_page'] == 'new_page')]
df2 = df2.append(df[(df['group'] == 'control') & (df['landing_page'] == 'old_page')])
df2.reset_index
df2.head(3)
```

Out[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

In [9]:
```
# Double Check all of the correct rows were removed - this should be 0
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].shape[0]
```

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

Extracted rows to df2 where treatment does not match with new_page or control does not match with old_page.

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**?

In [10]:
```
unique_users = df2.user_id.unique()
print('Number of unique users in df2 dataset: ', len(unique_users))
```

```
Number of unique users in df2 dataset:  290584
```

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

In [11]:
```
print('Reapeated user_id in df2 dataset:', );
df2[df2['user_id'].duplicated() == True]
```

```
Reapeated user_id in df2 dataset:
```

Out[11]:

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

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

In [12]:
```
df.iloc[2893]
```

Out[12]:
```
user_id                             773192
timestamp       2017-01-14 02:55:59.590927
group                            treatment
landing_page                      new_page
converted                                0
Name: 2893, dtype: object
```

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

In [13]:
```
df2.drop(2893, inplace=True)
```

In [14]:
```
df2[df2['user_id'].duplicated() == True]['user_id'].size # to prove that duplicate value don't exist
```

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

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 [15]:
```
df2.head(2)
```

Out[15]:

	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

In [16]:
```
all_converted_individuals = df2.query('converted == 1')
probability = all_converted_individuals.shape[0] / df2.shape[0]
print('Probability of converting regardless of page : ', probability)
```

```
Probability of converting regardless of page :  0.11959708724499628
```

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

In [17]:
```
con_gr_df = df2.query('group == "control"')
pr_conv_gr = con_gr_df[con_gr_df['converted'] == 1].shape[0]/con_gr_df.shape[0]
print('Given that an individual was in the control group, the probability of converting :', pr_conv_gr)
```

```
Given that an individual was in the control group, the probability of converting : 0.1203863045004612
```

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

In [18]:
```
treat_gr_df = df2.query('group == "treatment"')
pr_treat_gr = treat_gr_df[treat_gr_df['converted'] == 1].shape[0]/treat_gr_df.shape[0]
print('Given that an individual was in the treatment group, the probability of converting :', pr_treat_g
```

```
Given that an individual was in the treatment group, the probability of converting : 0.118808065515105
64
```

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

In [19]:
```
pr_new_page = df2.query('landing_page == "new_page"').shape[0] / df2.shape[0]
print('Given that an individual was in the treatment group, the probability of converting', pr_new_page)
```

```
Given that an individual was in the treatment group, the probability of converting 0.5000619442226688
```

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

- **It seems from above that there is no enough evidence that the new treatment has lead users to more conversions. In fact, probability of converted `control` group and `treatment` groups are almost same. In fact, probability of converted individuals `control(pr_conv_gr)` even slightly higher than `treatment(pr_treat_gr)`**
- **50% of all users recieved `new_page` and well balanced.**

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.

- **NULL HYPOTHESIS**

$$H_0 : \pi_{old} - \pi_{new} \geq 0$$

- **ALTERNATIVE HYPOTHESIS**

$$H_1 : \pi_{old} - \pi_{new} < 0$$

π_{old} and π_{new} are the converted rate values for old pages and new pages, respectivley. The aove hypothesis observe that old hypothesis is better until new page proves to be exactly better at Type I error rate of 5%

`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 **conversion rate** for p_{new} under the null?

In [20]:
```
df2.head(3)
```

Out[20]:

	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

In [21]:
```
all_conversions = df2[df2['converted'] == 1]
p_new = all_conversions.shape[0] / df2.shape[0]
print('p_new under the null :', p_new)
```

```
p_new under the null : 0.11959708724499628
```

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

In [22]:
```
all_conversions = df2[df2['converted'] == 1]
p_old = all_conversions.shape[0] / df2.shape[0]
print('p_old under the null :', p_old)
```

```
p_old under the null : 0.11959708724499628
```

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

In [23]:
```
n_new = df2.query('group == "treatment"').count()[0]
n_new
```

Out[23]: 145310

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

In [24]:
```
n_old = df2.query('group == "control"').count()[0]
n_old
```

Out[24]:
```
145274
```

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

In [25]:
```
new_page_converted = np.random.choice([0,1], size = n_new, p = [p_new, 1-p_new])
new_page_converted
```

Out[25]:
```
array([1, 1, 1, ..., 1, 1, 1])
```

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

In [26]:
```
old_page_converted = np.random.choice([0,1], size = n_old, p = [p_old, 1-p_old])
old_page_converted
```

Out[26]:
```
array([1, 1, 1, ..., 1, 1, 1])
```

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

In [27]:
```
p_diff = new_page_converted.mean() - old_page_converted.mean()
'{:.5f}'.format(p_diff)
```

Out[27]:
```
'-0.00206'
```

h. Create 10,000 p_{new} - p_{old} values using the same simulation process you used in parts (a) through (g) above. Store all 10,000 values in a NumPy array called **p_diffs**.

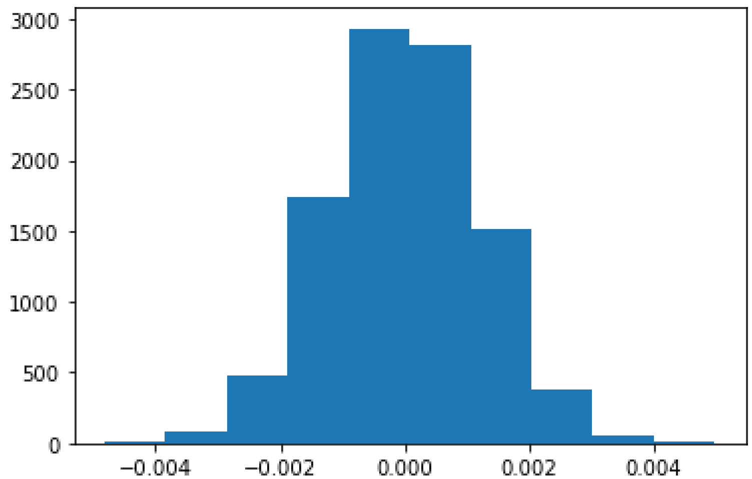
In [28]:
```
# p_diffs = []
# for _ in range(10000):
#     new_page_converted = np.random.choice([0,1], size = n_new, p = [p_new, 1-p_new])
#     old_page_converted = np.random.choice([0,1], size = n_old, p = [p_old, 1-p_old])
#     p_diffs.append(new_page_converted.mean() - old_page_converted.mean())
# p_diffs = np.array(p_diffs)
```

In [29]:
```
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
```

Out[29]:
```
array([ 0.00195243, -0.00148182, -0.00181899, ...,  0.00196652,
       -0.0016816 , -0.00080751])
```

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.

In [30]:
```
plt.hist(p_diffs);
```

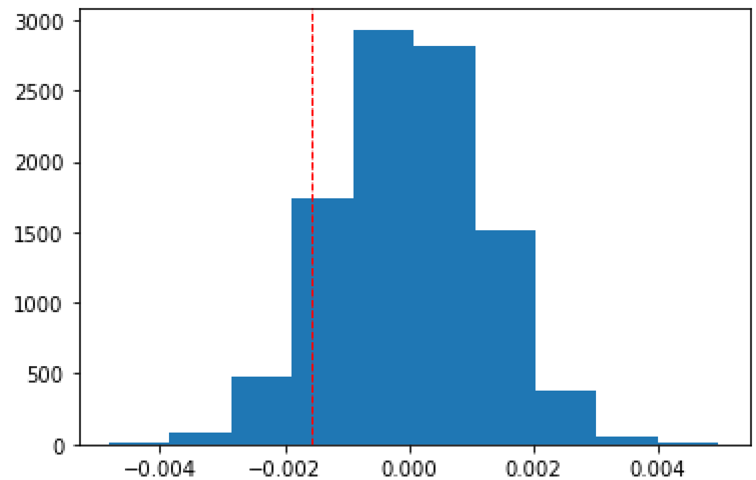


j. What proportion of the **p_diffs** are greater than the actual difference observed in **ab_data.csv**?

In [31]:
```
difference_ab_treatment = df2[df2['group'] == 'treatment']['converted'].mean()
difference_ab_control = df2[df2['group'] == 'control']['converted'].mean()
observed_difference = difference_ab_treatment - difference_ab_control
observed_difference
```

Out[31]:
```
-0.0015782389853555567
```

In [32]:
```
plt.hist(p_diffs);
plt.axvline(observed_difference, color='r', linestyle='dashed', linewidth=1);
```



In [33]:
```
#calculating pi value
(p_diffs > observed_difference).mean()
```

Out[33]: 0.9051

k. Please explain using the vocabulary you've learned in this course 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?

The difference in conversion rate is not statstically significant as pi-value is greater than or equal to 0.9. This means our pi-value is greater than threshold(alpha). So we fail to reject NULL HYPOTHESIS (H_0)

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.

In [34]:
```
import statsmodels.api as sm

convert_old = df2[df2['group'] == 'control']['converted'].sum()
convert_new = df2[df2['group'] == 'treatment']['converted'].sum()
n_old = df2[df2['landing_page'] == 'old_page'].shape[0]
n_new =  df2[df2['landing_page'] == 'new_page'].shape[0]
n_old, n_new, convert_old, convert_new
```

Out[34]:
```
(145274, 145310, 17489, 17264)
```

m. Now use `stats.proportions_ztest` to compute your test statistic and p-value. [Here (https://docs.w3cub.com/statsmodels/generated/statsmodels.stats.proportion.proportions_ztest/)](https://docs.w3cub.com/statsmodels/generated/statsmodels.stats.proportion.proportions_ztest/) is a helpful link on using the built in.

In [35]:
```
stat, pval = sm.stats.proportions_ztest([convert_old, convert_new][::-1], [n_old, n_new][::-1], alternat
stat, pval
```

Out[35]:
```
(-1.3109241984234394, 0.9050583127590245)
```

In [36]:
```
# for clarification after the review
print('z-score is from : ', observed_difference / p_diffs.std())
```

```
z-score is from :  -1.307160918413779
```

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 = -1.31 p-value = 0.9 this pi-value is not high from z-score at 95% confidence. So we fail to reject zero hypothesis (H_0)

Part III - A regression approach

`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 is either a conversion or no conversion, what type of regression should you be performing in this case?

Since response conversion is categorical data type(0 and 1), we should use Logistic reression

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 in df2 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.**

In [37]:
```
df2.head(2)
```

Out[37]:

	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

In [38]:
```
df2['intercept'] = 1
df2[['ab_page_0', 'ab_page_1']] = pd.get_dummies(df2['group'])
df2.head()
```

Out[38]:

	user_id	timestamp	group	landing_page	converted	intercept	ab_page_0	ab_page_1
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	0	1
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	0	1
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1	1	0	1
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1	1	0	1
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1	1	0	1

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

In [39]:
```
results = sm.Logit(df2['converted'], df2[['intercept', 'ab_page_1']]).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 [40]:
```
results.summary()
```

Out[40]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290582
Method:	MLE	**Df Model:**	1
Date:	Sun, 05 Sep 2021	**Pseudo R-squ.:**	8.077e-06
Time:	14:34:00	**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_1	-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 **Part II**?

Value $\pi_{value} = 0.19$ associated with ab_page_1(ab_page) lower than Part II ($\pi_{value} = 0.9$). This means the null and alternative hypohtesis are different each Parts.

- **NULL HYPOTHESIS**

$$H_0 : \pi_{old} - \pi_{new} = 0$$

- **ALTERNATIVE HYPOTHESIS**

$$H_1 : \pi_{old} - \pi_{new} \neq 0$$

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?

If we consider other things and add those values to the regression, it will show us relations between intercept and explanotory variable values, while holding variables constant

g. Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives in. You will need to read in the **countries.csv** dataset and merge together your datasets on the appropriate rows. [Here (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.join.html)](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.join.html) 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.

In [41]:
```
countries = pd.read_csv('countries.csv')
countries.head(3)
```

Out[41]:

	user_id	country
0	834778	UK
1	928468	US
2	822059	UK

In [42]:
```
df3 = pd.merge(countries, df2, on='user_id', how='inner')
print(df3['country'].unique())
df3.head()
```

```
['UK' 'US' 'CA']
```

Out[42]:

	user_id	country	timestamp	group	landing_page	converted	intercept	ab_page_0	ab_page_1
0	834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	1	0
1	928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	0	1
2	822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	0	1
3	711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	1	0
4	710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	0	1

In [43]:
```
df3 = df3.join(pd.get_dummies(df3['country']), how = 'right')
df3
```

Out[43]:

	user_id	country	timestamp	group	landing_page	converted	intercept	ab_page_0	ab_page_1	CA	UK	US
0	834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	1	0	0	1	0
1	928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	0	1	0	0	1
2	822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	0	1	0	1	0
3	711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	1	0	0	1	0
4	710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	0	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
290579	653118	US	2017-01-09 03:12:31.034796	control	old_page	0	1	1	0	0	0	1
290580	878226	UK	2017-01-05 15:02:50.334962	control	old_page	0	1	1	0	0	1	0
290581	799368	UK	2017-01-09 18:07:34.253935	control	old_page	0	1	1	0	0	1	0
290582	655535	CA	2017-01-09 13:30:47.524512	treatment	new_page	0	1	0	1	1	0	0
290583	934996	UK	2017-01-09 00:30:08.377677	control	old_page	0	1	1	0	0	1	0

290584 rows × 12 columns

In [44]:
```
y = df3['converted']
X = df3[['intercept', 'UK', 'US', 'ab_page_1']]
results = sm.Logit(y, X).fit()
```

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

In [45]:
```
results.summary()
```

Out[45]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290580
Method:	MLE	**Df Model:**	3
Date:	Sun, 05 Sep 2021	**Pseudo R-squ.:**	2.323e-05
Time:	14:34:01	**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
UK	0.0506	0.028	1.784	0.074	-0.005	0.106
US	0.0408	0.027	1.516	0.130	-0.012	0.093
ab_page_1	-0.0149	0.011	-1.307	0.191	-0.037	0.007

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.

In [46]:
```
df3['US_inter'] = df3['US'] * df3['ab_page_1']
df3['UK_inter'] = df3['UK'] * df3['ab_page_1']
y = df3['converted']
X = df3[['intercept', 'UK', 'UK_inter', 'US', 'US_inter', 'ab_page_1']]
results = sm.Logit(y, X).fit()
```

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

In [47]:
```
results.summary()
```

Out[47]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290578
Method:	MLE	**Df Model:**	5
Date:	Sun, 05 Sep 2021	**Pseudo R-squ.:**	3.482e-05
Time:	14:34:02	**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
UK	0.0118	0.040	0.296	0.767	-0.066	0.090
UK_inter	0.0783	0.057	1.378	0.168	-0.033	0.190
US	0.0175	0.038	0.465	0.642	-0.056	0.091
US_inter	0.0469	0.054	0.872	0.383	-0.059	0.152
ab_page_1	-0.0674	0.052	-1.297	0.195	-0.169	0.034

As p-value is higher than 0.05 for all variables with interactions between page and country, we can say there is no statistically signifanct effects on conversion.

Conclusion

Overall, study and results shows that there is no enough evidence to conversion to new_page than the old_page. Of course, considering other variable can make diffrence to conversion. However, based on my current results, I advise to not spend time much focus to build new page as it not gaining to much result.

Finishing Up

> Congratulations! You have reached the end of the A/B Test Results project! You should be very proud of all you have accomplished!

> **Tip**: Once you are satisfied with your work here, check over your report to make sure that it is satisfies all the areas of the rubric (found on the project submission page at the end of the lesson). You should also probably remove all of the "Tips" like this one so that the presentation is as polished as possible.

Directions to Submit

> Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).

> Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.

> Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

In [48]:
```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

Out[48]: 1

In [ ]: