Project 3: Analyze A/B Test Results

Yu Tao 12/19/2020

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
- Part I - Probability
- Part II - A/B Test
- Part III - Regression

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.

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)

import warnings
warnings.filterwarnings('ignore')
```

`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]:
```
# import the ab_data as df
df = pd.read_csv('ab_data.csv')

# print the first few lines of the dataframe
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 below cell to find the number of rows in the dataset.

In [3]:
```
# there are 294478 rows in the dataframe, we can check using info()
df.info()
```

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

c. The number of unique users in the dataset.

In [4]:
```
# count unique numbers in user_id column
df.user_id.nunique()
```

Out[4]: 290584

d. The proportion of users converted.

In [5]:
```
# calculate the mean value of the converted column
df.converted.mean()
```

Out[5]: 0.11965919355605512

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

In [6]:
```
# we can groupby 'group' and 'landing_page'
df.groupby(['group', 'landing_page']).count()
```

Out[6]:

		user_id	timestamp	converted
group	**landing_page**			
control	new_page	1928	1928	1928
	old_page	145274	145274	145274
treatment	new_page	145311	145311	145311
	old_page	1965	1965	1965

In [7]:
```
# sum over numbers of "control, new_page" and "treatment, old_page"
# this corresponds to the number of mismatched rows
1928 + 1965
```

Out[7]: 3893

f. Do any of the rows have missing values?

In [8]:
```
# from the code below, no line has missing values.
df.isnull().sum()
```

Out[8]:
```
user_id         0
timestamp       0
group           0
landing_page    0
converted       0
dtype: int64
```

`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**.

For the rows where treatment is not aligned with new_page or control is not aligned with old_page, we should remove those rows, we should only use the rows that we can feel confident in the accuracy of the data.

In [9]:
```
# query only for the "correct" rows
df2 = df.query('(group == "treatment" and landing_page == "new_page") or (group == "control" and landing_page == "ol
d_page")')
```

In [10]:
```
# check mismatched rows are all removed
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].shape[0]
```

Out[10]: 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**?

In [11]:
```
# count unique numbers in user_id column
df2.user_id.nunique()
```

Out[11]: 290584

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

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

Out[12]:

	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 [13]:
```
df2[df2['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**, but keep your dataframe as **df2**.

In [14]:
```
df2.drop([1899], inplace=True)
df2[df2['user_id'] == 773192]
```

Out[14]:

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

In [15]:
```
df2.converted.mean()
```

Out[15]: 0.11959708724499628

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

In [16]:
```
df2.query("group == 'control'").converted.mean()
```

Out[16]: 0.1203863045004612

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

In [17]:
```
df2.query("group == 'treatment'").converted.mean()
```

Out[17]: 0.11880806551510564

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

In [18]:
```
df2[df2['landing_page'] == 'new_page'].shape[0] / df2.shape[0]
```

Out[18]: 0.5000619442226688

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 isn't sufficient evidence to say that the new treatment page leads to more conversions. The reason is, even though the probability for a user to receive the old page or the new page is well controlled with a rough 50/50 chance, the conversion ratio in both groups are close (12.04% vs 11.88%), which makes it difficult to conclude on this treatment effect without any further evidence.

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.

The null hypothesis H_0: p_{new} <= p_{old}
The alternative hypothesis H_1: p_{new} > p_{old}

`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?

In [19]:
```
p_new = df2.converted.mean()
p_new
```

Out[19]: 0.11959708724499628

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

In [20]:
```
p_old = df2.converted.mean()
p_old
```

Out[20]: 0.11959708724499628

c. What is n_{new}?

In [21]:
```
n_new = df2.query("group == 'treatment'").user_id.nunique()
n_new
```

Out[21]: 145310

d. What is n_{old}?

In [22]:
```
n_old = df2.query("group == 'control'").user_id.nunique()
n_old
```

Out[22]: 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**.

In [23]:
```
new_page_converted = np.random.choice([1,0], size = n_new, p = (p_new, 1-p_new))
```

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**.

In [24]:
```
old_page_converted = np.random.choice([1,0], size = n_old, p = (p_old, 1-p_old))
```

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

In [25]:
```
# calculate pnew - pold
new_page_converted.mean() - old_page_converted.mean()
```

Out[25]: -0.0003807069393165913

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**.

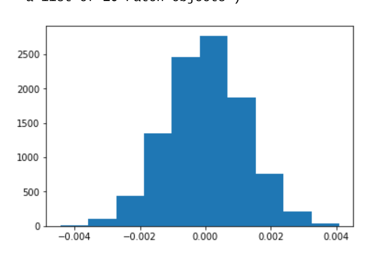
In [26]:
```
p_diffs = []

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

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 [27]:
```
plt.hist(p_diffs)
```

Out[27]:
```
(array([  17.,  104.,  435., 1344., 2453., 2768., 1868.,  762.,  211.,
          38.]),
 array([-0.00443452, -0.00358173, -0.00272893, -0.00187614, -0.00102335,
        -0.00017056,  0.00068224,  0.00153503,  0.00238782,  0.00324061,
         0.00409341]),
 <a list of 10 Patch objects>)
```



The above plot is the simulated sample distribution of Pnew-Pold, when the null hypothesis is true, therefore, it is approximately a normal distribution centered at 0 (no difference).

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

In [28]:
```
# first we calculate the actual difference
p_old_actual = df2.query("group == 'control'").converted.mean()
p_new_actual = df2.query("group == 'treatment'").converted.mean()
actual_diff = p_new_actual - p_old_actual
actual_diff
```

Out[28]: -0.0015782389853555567

In [29]:
```
# use the simulated data to create a normal distribution
p_diffs = np.array(p_diffs)
null_vals = np.random.normal(0, p_diffs.std(), p_diffs.size)

# the proportion of values on the bell curve that are greater than the actual difference
(null_vals > actual_diff).mean()
```

Out[29]: 0.9038

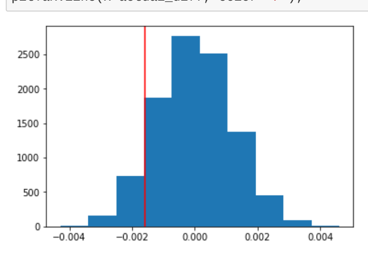
k. In words, explain what you just computed in part **j.** What is this value called in scientific studies? What does this value mean in terms of whether or not there is a difference between the new and old pages?

In part j, we calculated the p-value. The p-value is the probability of getting the observed statistic if the null hypothesis is true.

In a hypothesis test, we usually use a critical p-value: p = *0.05. If we obtain a very small p-value p* < p, it means that such an extreme observed outcome would be very unlikely under the null hypothesis. In our case, since here we have a p-value of 0.904, we fail to reject the null hypothesis. It means we can not tell there is any improvement in conversion rate using the new webpage.

Below, I use a vertical red line to indicate the actual conversion difference, the value of 0.904 we obtained before means that the probability of observing a higher conversion difference when the nul hypothesis is true is 90.4%.

In [30]:
```
plt.hist(null_vals);
plt.axvline(x=actual_diff, color='r');
```



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 [31]:
```
import statsmodels.api as sm

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

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

In [32]:
```
z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new], alternative='smaller')
z_score, p_value
```

Out[32]: (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.**?

Z-score (the standard score) can be computed from the formula Z=(x-μ)/σ, where x is the observed value, μ is the sample mean and σ is the standard deviation of the samples. In statistics, Z-score is the number of standard deviations by which the value of a raw score is above or below the mean value of what is being observed or measured. Usually with an alpha value of 0.05 the critical Z-score is 1.64. The actual Z-Score in this Z-test is smaller than the critical Z-score, which means we fail to reject the null hypothesis.

For the p-value of 0.905, it agrees with our findings in part j, therefore, we fail to reject the null.

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?

We need to perform a logistic regression, becuase the conversion column is a binary dependent 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**.

In [33]:
```
# recall what df2 looks like
df2.head()
```

Out[33]:

	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

In [34]:
```
# add intercept column
df2['intercept'] = 1

# add dummy variable
df2 = df2.join(pd.get_dummies(df2['group']))
df2.head()
```

Out[34]:

	user_id	timestamp	group	landing_page	converted	intercept	control	treatment
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	1	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	1	0
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
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	1	0

In [35]:
```
# rename treatment column and remove control column
df2.rename(columns = {'treatment': 'ab_page'}, inplace=True)
df2.drop('control', axis=1, inplace=True)
df2.head()
```

Out[35]:

	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

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.

In [36]:
```
# use statsmodels to do a logistic regression
x = df2[["intercept", "ab_page"]]
y = df2["converted"]

log_model = sm.Logit(y,x)
log_reg = 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.

In [37]:
```
log_reg.summary()
```

Out[37]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290582
Method:	MLE	**Df Model:**	1
Date:	Sat, 19 Dec 2020	**Pseudo R-squ.:**	8.077e-06
Time:	17:13:18	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		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

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**?

The p-value associated with ab_page is 0.190, which means we fail to reject the null.

It is different from the p-value we got earlier in part II because we are using different null and alternative hypotheses for part II and the logistic regression.

Here in the logistic regression model, the null/alternative hypothesis is that there isn't/is significant difference between the conversion rates of old/new webpage. Therefore, the p-value here is associated with a two-sided test (critical regions are 0.025 on each side of the sample distribution).

However, in part II, the null hypothesis is that the old webpage is better than the new webpage in terms of the conversion rate, therefore, the hypothesis test in part II is one-sided (with an alpha value of 0.05 on the right tail of the normal distribution).

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?

Other factors that might influence whether an individual converts include: the time of the day (morning, noon, afternoon, evening, night), the day of the week (weekdays, weekends) the individual sees the webpage (we can get this information from the time stamp column), the nation the individual comes from (it might also be associated with that person's income level), the gender of the individual, etc. Considering these factors might result in a more accurate model, and also help us describe the correlations between variables better. However, there are disadvantages that we over-complicate the model (such as adding quadratic or higher order terms in the regression), in this case the result might become difficult to interpret.

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.

In [38]:
```
# add country info to the dataframe
countries_df = pd.read_csv('./countries.csv')
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
df_new.head()
```

Out[38]:

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

In [39]:
```
# create the necessary dummy variables
df_new = df_new.join(pd.get_dummies(df_new['country']))
df_new.head()
```

Out[39]:

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

In [40]:
```
# logistic regression fitting and summary

x = df_new[["intercept", "ab_page", "CA", "UK"]]
y = df_new["converted"]

log_model_2 = sm.Logit(y,x)
log_reg_2 = log_model_2.fit()
log_reg_2.summary()
```

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

Out[40]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290580
Method:	MLE	**Df Model:**	3
Date:	Sat, 19 Dec 2020	**Pseudo R-squ.:**	2.323e-05
Time:	17:13:20	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.1760

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-1.9893	0.009	-223.763	0.000	-2.007	-1.972
ab_page	-0.0149	0.011	-1.307	0.191	-0.037	0.007
CA	-0.0408	0.027	-1.516	0.130	-0.093	0.012

Based on the p-values (0.191, 0.130, 0.457) associated with the page and country features, which are all greater than 0.050, we conclude that old/new webpages and countries where the individual lives don't have a significant impact on the conversion rate.

Based on the coef column, we can also calculate their reciprocal expontials and get some extra conclusions.

In [41]:
```
1/np.exp(-0.0149), 1/np.exp(-0.0408), 1/np.exp(0.0099)
```

Out[41]: (1.0150115583846535, 1.0416437559600236, 0.9901488436829571)

If the individual receives the old webpage, he/she is 1.02 times more likely to convert.

If the individual is currently living in US, he/she is 1.04 times more likely to convert than an individual living in Canada.

If the individual is currently living in US, he/she is 0.99 times more likely (by which I mean less likely)to convert than an individual living in UK.

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 are 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 [42]:
```
# recheck the dataframe
df_new.head()
```

Out[42]:

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

In [43]:
```
# define two interaction terms, one for CA and ab_page, one for UK and ab_page
df_new['CA_ab_page'] = df_new['CA'] * df_new['ab_page']
df_new['UK_ab_page'] = df_new['UK'] * df_new['ab_page']

df_new.head()
```

Out[43]:

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

In [44]:
```
# logistic regression model and summary

x = df_new[["intercept", "ab_page", "CA", "UK", "CA_ab_page", "UK_ab_page"]]
y = df_new["converted"]

log_model_3 = sm.Logit(y,x)
log_reg_3 = log_model_3.fit()
log_reg_3.summary()
```

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

Out[44]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290578
Method:	MLE	**Df Model:**	5
Date:	Sat, 19 Dec 2020	**Pseudo R-squ.:**	3.482e-05
Time:	17:13:22	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.1920

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-1.9865	0.010	-206.344	0.000	-2.005	-1.968
ab_page	-0.0206	0.014	-1.505	0.132	-0.047	0.006
CA	-0.0175	0.038	-0.465	0.642	-0.091	0.056
UK	-0.0057	0.019	-0.306	0.760	-0.043	0.031
CA_ab_page	-0.0469	0.054	-0.872	0.383	-0.152	0.059

Based on the p-values associated with the interaction terms (0.383, 0.238), they are both larger than 0.05, which means these higher order terms are not statistically significant.

Conclusions

Congratulations on completing the project!

Gather Submission Materials

Once you are satisfied with the status of your Notebook, you should save it in a format that will make it easy for others to read. You can use the **File -> Download as -> HTML (.html)** menu to save your notebook as an .html file. If you are working locally and get an error about "No module name", then open a terminal and try installing the missing module using `pip install <module_name>` (don't include the "<" or ">" or any words following a period in the module name).

You will submit both your original Notebook and an HTML or PDF copy of the Notebook for review. There is no need for you to include any data files with your submission. If you made reference to other websites, books, and other resources to help you in solving tasks in the project, make sure that you document them. It is recommended that you either add a "Resources" section in a Markdown cell at the end of the Notebook report, or you can include a `readme.txt` file documenting your sources.

Submit the Project

When you're ready, click on the "Submit Project" button to go to the project submission page. You can submit your files as a .zip archive or you can link to a GitHub repository containing your project files. If you go with GitHub, note that your submission will be a snapshot of the linked repository at time of submission. It is recommended that you keep each project in a separate repository to avoid any potential confusion: if a reviewer gets multiple folders representing multiple projects, there might be confusion regarding what project is to be evaluated.

It can take us up to a week to grade the project, but in most cases it is much faster. You will get an email once your submission has been reviewed. If you are having any problems submitting your project or wish to check on the status of your submission, please email us at dataanalyst-project@udacity.com. In the meantime, you should feel free to continue on with your learning journey by beginning the next module in the program.

In [ ]: