Analyze_ab_test_results_notebook_Austin

May 22, 2021

0.1 Analyze A/B Test Results: New vs Old Page

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. **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!

0.2 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.

Part I - Probability

To get started, let's import our libraries.

```
In [222]: 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 [223]: df = pd.read_csv('ab_data.csv')
          df.head()
```

```
Out[223]:    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 [224]: df.shape[0]
```

```
Out[224]: 294478
```

c. The number of unique users in the dataset.

```
In [225]: df.user_id.nunique()
```

```
Out[225]: 290584
```

d. The proportion of users converted.

```
In [226]: df['converted'].sum()/290584
```

```
Out[226]: 0.12126269856564711
```

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

```
In [227]: no_lineup1 = df.query("group == 'treatment' and landing_page == 'old_page'").shape[0]
          no_lineup2 = df.query("group == 'control' and landing_page == 'new_page'").shape[0]

          no_lineup1 + no_lineup2
```

```
Out[227]: 3893
```

f. Do any of the rows have missing values?

```
In [228]: 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
```

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 [229]: df2 = df.query("group == 'control' and landing_page == 'old_page'")
          df2 = df2.append(df.query("group == 'treatment' and landing_page == 'new_page'"))
```

```
In [230]: # Checkimg all of the proper rows were removed
          df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].s
```

```
Out[230]: 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 [231]: df2.user_id.nunique()
```

```
Out[231]: 290584
```

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

```
In [232]: df2[df2['user_id'].duplicated()]
```

```
Out[232]:       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 [233]: df2[df2['user_id'] == 773192]
```

```
Out[233]:       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 [234]: df2 = df2.drop(1899)
```

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 [235]: df2.converted.mean()
```

```
Out[235]: 0.11959708724499628
```

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

```
In [236]: C_prob = df2.query("group == 'control'").converted.mean()
```

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

```
In [237]: T_prob = df2.query("group == 'treatment'").converted.mean()
```

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

```
In [238]: df2.query("landing_page == 'new_page'").shape[0] / df2.landing_page.shape[0]
```

```
Out[238]: 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.

No, There is no significant evidence for us to conclude that the new treatment page leads to more conversions. Both groups conclude at an almost identical average of 12%. This allows us to conclude that their is no significant evidence for a change in conversion with from old to new landing page.

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.

H0:Pold=Pnew H1:Pnew>Pold or H0:PoldPnew=0 H1:PnewPold>0

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

Out[239]: 0.11959708724499628
```

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

```
In [240]: p_old = df2.converted.mean()
          p_old

Out[240]: 0.11959708724499628
```

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

```
In [241]: n_new = df2.query("group == 'treatment'").shape[0]
          n_new

Out[241]: 145310
```

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

```
In [242]: n_old = df2.query("group == 'control'").shape[0]
          n_old

Out[242]: 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 [243]: new_page_converted = np.random.choice([0, 1], size = n_new, p = [p_new, 1 - p_new])
          new_page_converted

Out[243]: 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 [244]: old_page_converted = np.random.choice([0, 1], size = n_old, p = [p_old, 1 - p_old])
          old_page_converted

Out[244]: array([1, 1, 1, ..., 1, 0, 0])
```

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

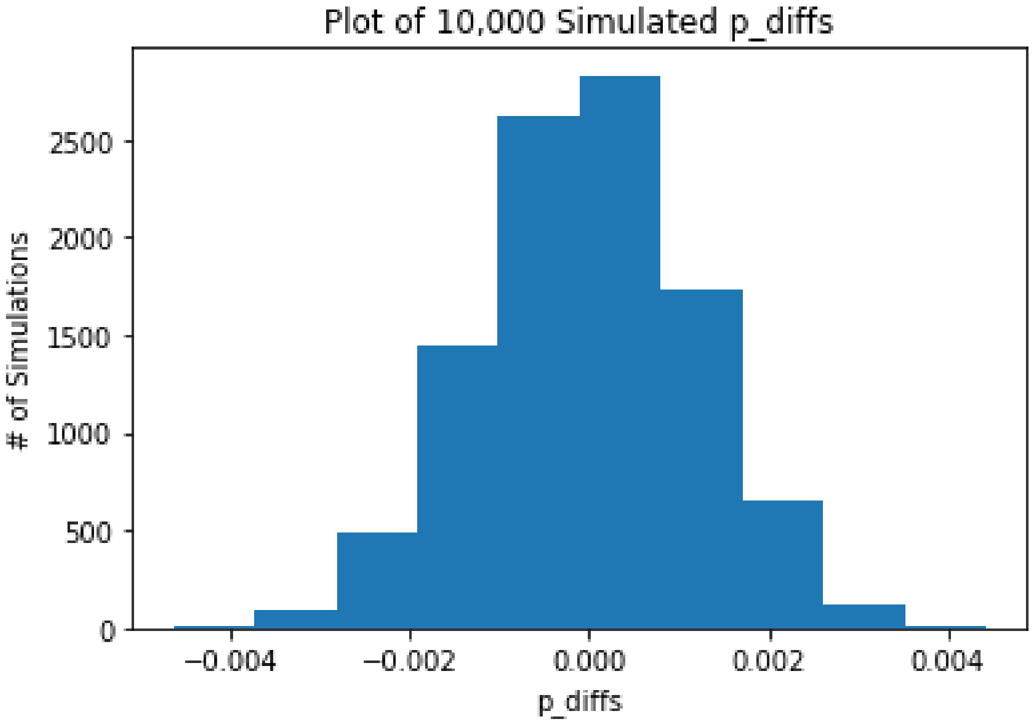
```
In [245]: observ_diff= new_page_converted.mean() - old_page_converted.mean()
          observ_diff

Out[245]: 9.1811254436469092e-05
```

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 [246]: p_diffs = []
          new_conv_simulation = np.random.binomial(n_new, p_new, 10000)/n_new
          old_conv_simulation = np.random.binomial(n_old, p_new, 10000)/n_old
          p_diffs = new_conv_simulation - old_conv_simulation
```

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 [247]: plt.hist(p_diffs);
          plt.ylabel('# of Simulations')
          plt.xlabel('p_diffs')
          plt.title('Plot of 10,000 Simulated p_diffs');
```

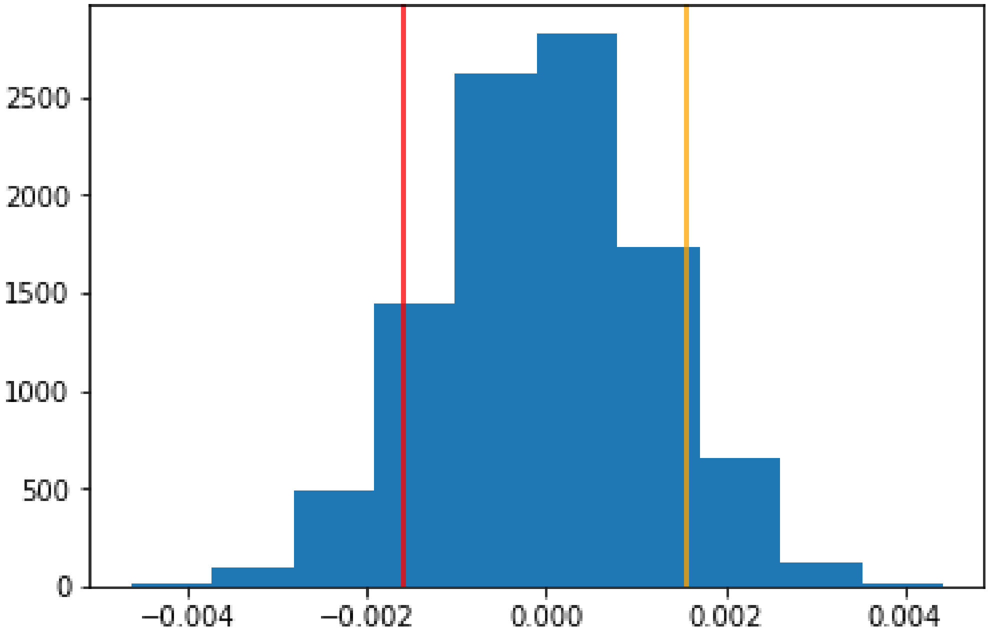


j. What proportion of the **p_diffs** are greater than the actual difference observed in **ab_data.csv**?

```
In [248]: obs_diff = T_prob - C_prob

          low_prob = (p_diffs < obs_diff).mean()
          high_prob = (p_diffs.mean() + (p_diffs.mean() - obs_diff) < p_diffs).mean()
```

```
plt.hist(p_diffs);
plt.axvline(obs_diff, color='red');
plt.axvline(p_diffs.mean() + (p_diffs.mean() - obs_diff), color='orange');

#proportion of the p_diffs are greater than the actual difference observed in ab_data.

(p_diffs > obs_diff).mean()
```

```
Out[248]: 0.90069999999999995
```



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 value 0.9058 we call it P-value, which proposes if there is a compelling difference between 2 groups. In this case, the new page doesn't have better conversion rates than the old page because the value 0.9 is much higher than the alpha(0.05) which means we do not have evidence to reject the null hypothesis, so we fail to reject the null.

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 [249]: convert_old = df2.query("landing_page == 'old_page'")['converted'].sum()
          convert_new = df2.query("landing_page == 'new_page'")['converted'].sum()
```

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 [250]: import statsmodels.api as sm

          z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_ne
          z_score, p_value

Out[250]: (1.3109241984234394, 0.90505831275902449)
```

```
In [251]: #import the norm function to compute the significance of our z-score.
          from scipy.stats import norm

          norm.cdf(z_score)

Out[251]: 0.90505831275902449
```

```
In [252]: #Next we check our critical value at 95% confidence interval.
          norm.ppf(1-(0.05/2))

Out[252]: 1.959963984540054
```

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: 0.905 We know that the z-score reffers to the difference between our test statistic or the difference between conversion rates. The null hypothesis is 1.31 Standard deviations above the mean. To reject the null we would need a value above 1.96, but we have 1.31<1.96. We also know that our p-value is 0.905 which is > than the alpha of 0.05. The p-value determines the significance of our resuls. The values are different from parts j and k but similarly still suggests no statistically significant difference betweem the new and the old page. These findings fail to reject the null hypothesis.

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?

Logistic Regression, response variable is categorical

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 [253]: # create a colun for the intercept
          df2['intercept'] = 1
          df2.head()

Out[253]:    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
          4   864975  2017-01-21 01:52:26.210827  control     old_page          1
          5   936923  2017-01-10 15:20:49.083499  control     old_page          0
          7   719014  2017-01-17 01:48:29.539573  control     old_page          0

             intercept
          0          1
          1          1
          4          1
          5          1
          7          1
```

```
In [254]: # create a dummy variable column for which page each user received
          df2['ab_page'] = pd.get_dummies(df['group'])['treatment']
          df2.head()

Out[254]:    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
          4   864975  2017-01-21 01:52:26.210827  control     old_page          1
          5   936923  2017-01-10 15:20:49.083499  control     old_page          0
          7   719014  2017-01-17 01:48:29.539573  control     old_page          0

             intercept  ab_page
          0          1        0
          1          1        0
          4          1        0
          5          1        0
          7          1        0
```

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 [255]: log_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
          results = log_mod.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 [256]: results.summary2()
```

```
Out[256]: <class 'statsmodels.iolib.summary2.Summary'>
          """
                                    Results: Logit
          ==================================================================
          Model:              Logit            No. Iterations:   6.0000
          Dependent Variable: converted        Pseudo R-squared: 0.000
          Date:               2021-05-22 22:19 AIC:              212780.3502
          No. Observations:   290584           BIC:              212801.5095
          Df Model:           1                Log-Likelihood:   -1.0639e+05
          Df Residuals:       290582           LL-Null:          -1.0639e+05
          Converged:          1.0000           Scale:            1.0000
          -------------------------------------------------------------------
                        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**? **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**?

The p-value associated with ab_page was 0.189 which was significantly lower than the result in Part II which was 0.905. The reason for such a significant difference is because the null and alternative hypothesis differed in each exercise.

H0: pold - pnew >(or equal to) 0 H1: pold -pnew < 0

H0: pold = pnew H1: pold not equal to pnew

Why does it differ from the value you found in Part II? Because the later case relies solely on two possible outcomes.

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?

It is good idea to consider other factors to into the regression model because the treatment/control page does not have significant impact on user conversion. When considering other factors we should check to make sure they are not colinear.

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 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 [257]: # Store Countries.csv data in dataframe
          countries = pd.read_csv('countries.csv')
          countries.head()
```

```
Out[257]:    user_id country
          0   834778      UK
          1   928468      US
          2   822059      UK
          3   711597      UK
          4   710616      UK
```

```
In [258]: #Inner join two datas
          new = countries.set_index('user_id').join(df2.set_index('user_id'), how = 'inner')
          new.head()
```

```
Out[258]:        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
          user_id
          834778           0          1        0
          928468           0          1        1
          822059           1          1        1
          711597           0          1        0
          710616           0          1        1
```

```
In [259]: #adding dummy variables with 'CA' as the baseline
          new[['US', 'UK']] = pd.get_dummies(new['country'])[['US', "UK"]]
          new.head()
```

```
Out[259]:        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  US  UK
          user_id
          834778           0          1        0   0   1
          928468           0          1        1   1   0
          822059           1          1        1   0   1
          711597           0          1        0   0   1
          710616           0          1        1   0   1
```

```
In [260]: new['US_ab_page'] = new['US']*new['ab_page']
          new.head()

Out[260]:        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  US  UK  US_ab_page
          user_id
          834778           0          1        0   0   1           0
          928468           0          1        1   1   0           1
          822059           1          1        1   0   1           0
          711597           0          1        0   0   1           0
          710616           0          1        1   0   1           0

In [261]: new['UK_ab_page'] = new['UK']*new['ab_page']
          new.head()

Out[261]:        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  US  UK  US_ab_page  UK_ab_page
          user_id
          834778           0          1        0   0   1           0           0
          928468           0          1        1   1   0           1           0
          822059           1          1        1   0   1           0           1
          711597           0          1        0   0   1           0           0
          710616           0          1        1   0   1           0           1

In [262]: logit3 = sm.Logit(new['converted'], new[['intercept', 'ab_page', 'US', 'UK', 'US_ab_pa
          logit3

Out[262]: <statsmodels.discrete.discrete_model.Logit at 0x7f7e5bca7f28>

In [263]: #Check the result
          result3 = logit3.fit()

Optimization terminated successfully.
         Current function value: 0.366112
         Iterations 6
```

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 [264]: result3.summary2()
```

```
Out[264]: <class 'statsmodels.iolib.summary2.Summary'>
"""
                                Results: Logit
========================================================================
Model:                 Logit              No. Iterations:     6.0000
Dependent Variable:    converted          Pseudo R-squared:   0.000
Date:                  2021-05-22 22:19   AIC:                212782.5603
No. Observations:      290584             BIC:                212835.4585
Df Model:              4                  Log-Likelihood:     -1.0639e+05
Df Residuals:          290579             LL-Null:            -1.0639e+05
Converged:             1.0000             Scale:              1.0000
------------------------------------------------------------------------
             Coef.    Std.Err.     z     P>|z|      [0.025       0.975]
------------------------------------------------------------------------
intercept  -2.0366      0.0280 -72.6176 0.0000       -2.0916      -1.9817
ab_page    -0.0018      0.0209  -0.0861 0.9313       -0.0427       0.0391
US          0.0501      0.0297   1.6912 0.0908       -0.0080       0.1083
UK          0.0507      0.0284   1.7860 0.0741       -0.0049       0.1064
US_ab_page -0.0094 704237.3061  -0.0000 1.0000 -1380279.7660 1380279.7472
US_ab_page -0.0094 704237.3061  -0.0000 1.0000 -1380279.7660 1380279.7472
========================================================================

"""
```

Conclusions: We see that through our p-values above, it does not seem that country has real significant impact on page coverstion rate. None of the variables have significant p-values. Therefore, we will fail to reject the null and conclude that there is not sufficient evidence to suggest that there is an interaction between country and page received that will predict whether a user converts or not. Big picture, based on the available information, we do not have sufficient evidence to suggest that the new page results provides more conversions than the old page.

0.3 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 [ ]:
```