Analyze_ab_test_results_notebook

July 24, 2021

0.1 Analyze A/B Test Results

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.

[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:

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

[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.

[3]:
```
df.shape # The shape of the dataset
```

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

c. The number of unique users in the dataset.

[4]:
```
# show the number of unique users
df['user_id'].nunique()
```

[4]:
```
290584
```

d. The proportion of users converted.

[5]:
```
# Proportion of users they're converted

print("The proportion is {0:.2%}".format(df['converted'].mean())) #␣
 ↪Approximately 12%
```

```
The proportion is 11.97%
```

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

[6]:
```
# Create two dataframe to get count number of each new page and treatment
df1 = df.loc[ (df['group'] != 'treatment') & (df['landing_page'] ==␣
 ↪'new_page')].count()

df2 = df.loc[ ( df['group'] == 'treatment') & (df['landing_page'] !=␣
 ↪'new_page') ].count()

print(df1)
print('_________________________')
print(df2)
```

```
user_id         1928
timestamp       1928
group           1928
landing_page    1928
converted       1928
dtype: int64
------------------------
user_id         1965
timestamp       1965
group           1965
landing_page    1965
converted       1965
dtype: int64
```

[7]:
```
# sum two numbers above to get the number of times the new_page and treatment don't line up.
1928 + 1965
```

[7]:
```
3893
```

f. Do any of the rows have missing values?

[8]:
```
#check if there is any null and missing values

df.isnull().sum().any()
```

[8]:
```
False
```

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**.

[9]:
```
# drop rows that treatment not aligned with new page or control is not aligned with old page

df2 = df.drop(df.query('(group == "treatment" and landing_page != "new_page") or (group != "treatment" and landing_page == "new_page") or (group == "control" and landing_page != "old_page") or (group != "control" and landing_page == "old_page")').index)
df2
```

[9]:

	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
…         …                          …        …           …        …
294473   751197  2017-01-03 22:28:38.630509    control     old_page          0
294474   945152  2017-01-12 00:51:57.078372    control     old_page          0
294475   734608  2017-01-22 11:45:03.439544    control     old_page          0
294476   697314  2017-01-15 01:20:28.957438    control     old_page          0
294477   715931  2017-01-16 12:40:24.467417  treatment     new_page          0

[290585 rows x 5 columns]
```

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**?

[10]:
```
# show number of unique users
df2['user_id'].nunique()
```

[10]:
```
290584
```

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

[11]:
```
# Show the duplicated user id in df2

df2[df2['user_id'].duplicated()]['user_id']
```

[11]:
```
2893    773192
Name: user_id, dtype: int64
```

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

[12]:
```
# Show the information of duplicated 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**, but keep your dataframe as **df2**.

[13]:
```
# drop duplicated user id
df2['user_id'].drop_duplicates(keep=False, inplace=True)
```

[14]:
```
# check if still there is a duplicated user id
df2['user_id'].duplicated().sum()
```

[14]:
```
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?

[15]:
```
# show the proportion of probability that users converted

print("The probability is {0:.2%}".format(df2['converted'].mean())) #␣
 ↪Approximately 12%
```

```
The probability is 11.96%
```

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

[16]:
```
# show the proportion of probability that users converted with control group

p = float(df2.query('group == "control"')['converted'].mean())
print("The probability is {0:.2%}".format(p))
```

```
The probability is 12.04%
```

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

[17]:
```
# show the proportion of probability that users converted with treatment group

p1 = float(df2.query('group == "treatment"')['converted'].mean())
print("The probability is {0:.2%}".format(p1))
```

```
The probability is 11.88%
```

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

[18]:
```
# show the proportion of probability that Individual received the new page

p2 = float(df2.query('landing_page == "new_page"')['user_id'].nunique())/ df2.
 ↪shape[0]
print("The probability is {0:.2%}".format(p2))
```

```
The probability is 50.01%
```

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.

From our findings above and regardless of page type the probability of the control group is 12.04% and the probability of the treatment group is 11.88% as we can see the treatment is less than the control group though we didn't take any consideration about page type if it is old or new. So, I think the control group is better in a little bit than the treatment group.

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.

Hypothesis:

Null Hypothesis (H_0): $p_{old} >= p_{new}$ Assume the old page is better.

Alternative Hypothesis (H_1): $p_{old} < p_{new}$ Prove the new page is better.

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?

[19]:
```
# Show convert rate for P_new
P_new = df2['converted'].mean()
P_new
```

[19]:
```
0.11959667567149027
```

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

[20]:
```
# Show convert rate for P_old
P_old = df2['converted'].mean()
P_old
```

[20]:
```
0.11959667567149027
```

c. What is n_{new}?

[21]:
```
# Calculate number of using new page
n_new = df2.query("landing_page == 'new_page'")['user_id'].nunique()
n_new
```

[21]:
```
145310
```

d. What is n_{old}?

[22]:
```
# Calculate number of using old page

n_old = len(df2.query("landing_page == 'old_page'"))
n_old
```

[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**.

[23]:
```
#Simulate n_new transactions
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**.

[24]:
```
#Simulate n_old transactions
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).

[25]:
```
new_page_converted.mean() - old_page_converted.mean()
```

[25]:
```
-0.0015301684358783735
```

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**.

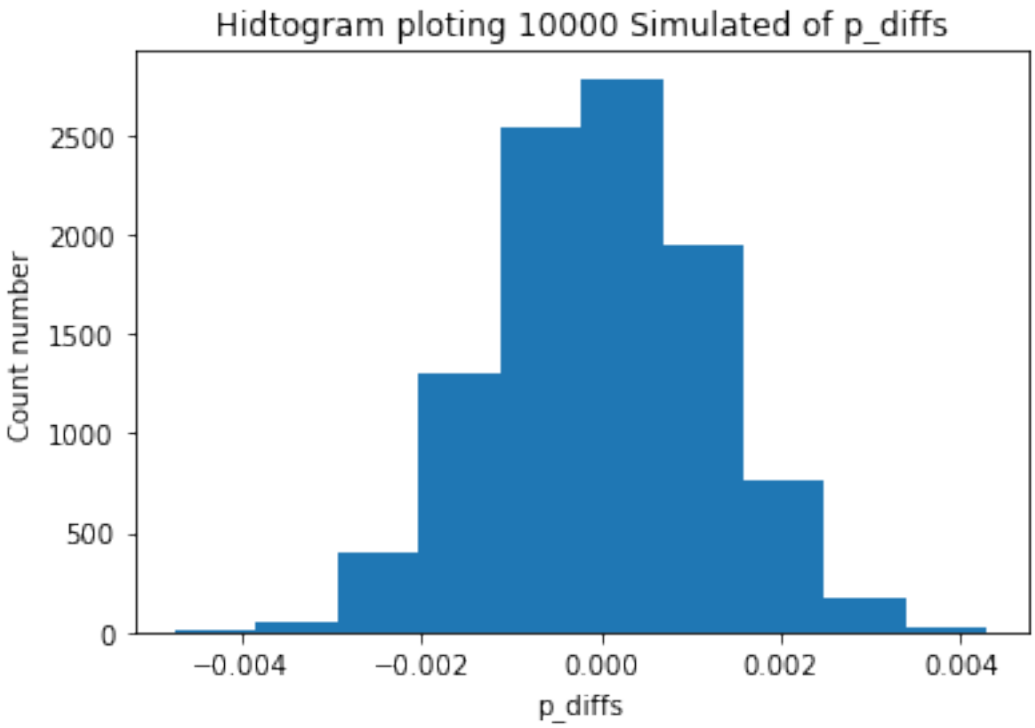
[27]:
```
# Simulate 10,000 P_new - P_old

p_diffs = []
new_converted_simulation = np.random.binomial(n_new, P_old, 10000)/n_new
old_converted_simulation = np.random.binomial(n_old, P_old, 10000)/n_old
p_diffs = new_converted_simulation - old_converted_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.

[28]:
```
#plot the p_diffs in histogram
plt.hist(p_diffs)
plt.xlabel('p_diffs')
plt.ylabel('Count number')
plt.title('Hidtogram ploting 10000 Simulated of p_diffs')
```

[28]:
```
Text(0.5, 1.0, 'Hidtogram ploting 10000 Simulated of p_diffs')
```



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

[29]:
```
#Calculate difference

actual_differ = df2.query('group == "treatment"')['converted'].mean() - df2.
 ↪query('group == "control"')['converted'].mean()
actual_differ
```

[29]:
```
-0.0015790565976871451
```

[30]:
```
# Show the proprtion of p_diffs are greater than the actual difference
(np.array(p_diffs) > actual_differ).mean()
```

[30]:
```
0.907
```

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?

The value called in scientific studies **P_value**. As shown above in part j the P_value is above 0.05 so, we do not have any evidence and we fail to reject the null hypothises.

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.

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


convert_old = df2.query("landing_page == 'old_page' and converted == 1").
 ↪shape[0]
convert_new = df2.query("landing_page == 'new_page' and converted == 1").
 ↪shape[0]
n_old = df2.query("landing_page == 'old_page'").shape[0]
n_new = df2.query("landing_page == 'new_page'").shape[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.

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

[32]:
```
(-1.3116075339133115, 0.905173705140591)
```

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.**?

The z-score is equal -1.31 and is the distance between an individual score and the mean in standard deviation units, also known as a standardized score. Also, the P_value is still large that means both of them are agree with our findings in part j and k so, we can't reject the null hypothesis.

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?

The Logistic Regression will be performed in this case.

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**.

[33]:
```
# add intercept column
df2['intercept']=1

# add ab_page column and get dummy variables
```

```
df2['ab_page'] = pd.get_dummies(df2['group'])['treatment']
```

[34]:
```
# show dataset

df2.head()
```

[34]:
```
   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

   intercept  ab_page
0          1        0
1          1        0
2          1        1
3          1        1
4          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.

[35]:
```
# using logistic model
log_m=sm.Logit(df2['converted'],df2[['intercept','ab_page']])
results=log_m.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.

[36]:
```
#Show summary
results.summary()
```

[36]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290585
Model:                          Logit   Df Residuals:                   290583
Method:                           MLE   Df Model:                            1
Date:                Sat, 24 Jul 2021   Pseudo R-squ.:               8.085e-06
Time:                        21:27:53   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
Covariance Type:            nonrobust   LLR p-value:                    0.1897
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9888      0.008   -246.669      0.000      -2.005      -1.973
ab_page       -0.0150      0.011     -1.312      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 the **Part II**?

The P-value associated with ab_page is 0.190 which is differ from value found in Part II beacause the Null Hypothesis (H_0): $p_{old} >= p_{new}$ and the Alternative Hypothesis (H_1): $p_{old} < p_{new}$.

While here in the Logistics Regression the Null Hypothesis (H_0): $p_{old} = p_{new}$ and the Alternative Hypothesis (H_1): p_{old} != p_{new}.

The difference between p-values of Part 2 and 3 is because we have performed a one-tailed test in Part 2, and in Part 3, we are performing a two-tailed test.

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?

There are other factors to be considerd such as timestamp that might be influenced on individual converts and also affect our decision-making and results. The disadvantages of adding additional terms into the regression model that makes the model explanation more complex as well as the combined impact of different variables disappears or reverses when these variables are combined, but appears where these variables are tested individually.

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.

[37]:
```
countries_df = pd.read_csv('./countries.csv')
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'),␣
 ↪how='inner')
```

[38]:
```
# Show dataset
df_new.head()
```

[38]:
```
        country                   timestamp      group landing_page  \
user_id
630000       US  2017-01-19 06:26:06.548941  treatment     new_page
630001       US  2017-01-16 03:16:42.560309  treatment     new_page
630002       US  2017-01-19 19:20:56.438330    control     old_page
```

```
630003       US  2017-01-12 10:09:31.510471  treatment     new_page
630004       US  2017-01-18 20:23:58.824994  treatment     new_page

         converted  intercept  ab_page
user_id
630000           0          1        1
630001           1          1        1
630002           0          1        0
630003           0          1        1
630004           0          1        1
```

[39]:
```
### Create the necessary dummy variables

df_new[['CA', 'UK', 'US']] = pd.get_dummies(df_new['country'])
df_new.head()
```

[39]:
```
        country                   timestamp      group landing_page  \
user_id
630000       US  2017-01-19 06:26:06.548941  treatment     new_page
630001       US  2017-01-16 03:16:42.560309  treatment     new_page
630002       US  2017-01-19 19:20:56.438330    control     old_page
630003       US  2017-01-12 10:09:31.510471  treatment     new_page
630004       US  2017-01-18 20:23:58.824994  treatment     new_page

         converted  intercept  ab_page  CA  UK  US
user_id
630000           0          1        1   0   0   1
630001           1          1        1   0   0   1
630002           0          1        0   0   0   1
630003           0          1        1   0   0   1
630004           0          1        1   0   0   1
```

[40]:
```
### Fit Your Linear Model And Obtain the Results
df_new['intercept'] = 1
log_model = sm.Logit(df_new['converted'], df_new[['intercept', 'ab_page', 'CA',␣
 ↪'UK']])
results = log_model.fit()
results.summary()
```

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

[40]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
```

```
Dep. Variable:                converted   No. Observations:                290585
Model:                          Logit   Df Residuals:                    290581
Method:                           MLE   Df Model:                             3
Date:                Sat, 24 Jul 2021   Pseudo R-squ.:               2.324e-05
Time:                        21:30:11   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
Covariance Type:            nonrobust   LLR p-value:                     0.1758
==============================================================================
                 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       -0.0150      0.011     -1.308      0.191      -0.037       0.007
CA            -0.0408      0.027     -1.516      0.130      -0.093       0.012
UK             0.0099      0.013      0.744      0.457      -0.016       0.036
==============================================================================
"""
```

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.

[41]:
```
df_new['US_ab_page'] = df_new['US']*df_new['ab_page']
df_new['UK_ab_page'] = df_new['UK']*df_new['ab_page']

df_new.head()
```

[41]:

user_id	country	timestamp	group	landing_page	converted	intercept	ab_page	CA	UK	US	US_ab_page	UK_ab_page
630000	US	2017-01-19 06:26:06.548941	treatment	new_page	0	1	1	0	0	1	1	0
630001	US	2017-01-16 03:16:42.560309	treatment	new_page	1	1	1	0	0	1	1	0
630002	US	2017-01-19 19:20:56.438330	control	old_page	0	1	0	0	0	1	0	0
630003	US	2017-01-12 10:09:31.510471	treatment	new_page	0	1	1	0	0	1	1	0
630004	US	2017-01-18 20:23:58.824994	treatment	new_page	0	1	1	0	0	1	1	0

[42]:
```
logit_model = sm.Logit(df_new['converted'], df_new[['intercept', 'ab_page',␣
↪'US', 'UK', 'US_ab_page', 'UK_ab_page']])
results = logit_model.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366108
         Iterations 6
```

[42]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290585
Model:                          Logit   Df Residuals:                   290579
Method:                           MLE   Df Model:                            5
Date:                Sat, 24 Jul 2021   Pseudo R-squ.:               3.483e-05
Time:                        21:39:55   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
Covariance Type:            nonrobust   LLR p-value:                    0.1918
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -2.0040      0.036    -55.008      0.000      -2.075      -1.933
ab_page       -0.0674      0.052     -1.297      0.195      -0.169       0.034
US             0.0175      0.038      0.465      0.642      -0.056       0.091
UK             0.0118      0.040      0.296      0.767      -0.066       0.090
US_ab_page     0.0469      0.054      0.872      0.383      -0.059       0.152
UK_ab_page     0.0783      0.057      1.378      0.168      -0.033       0.190
==============================================================================
"""
```

Conclusions

In this project, the goal was to prove which page leads to more conversion than another page. The null hypothesis assumed the old page has higher more conversion or the same as the new page. Also, the alternative page is to prove the new page leads more conversion than older page. We performed A/B Testing to get a result and the results showed that the factors of landing page and country do not lead to significant effect on the converted rate individually as well as interactively. So, we fail to reject The Null Hypothesis and reject The Alternative Hypothesis.

0.2.1 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.

0.2.2 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.

0.3 References

- https://verascity.github.io/ab_test.html
- https://www.youtube.com/watch?v=VuKIN9S8Ivs
- https://online.stat.psu.edu/stat200/lesson/5/5.1