

# Analyze\_ab\_test\_results\_notebook

November 1, 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

- Section ??
- Section ??
- Section ??
- Section ??

### ### 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.

```
[2]: 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:

```
[3]: df = pd.read_csv('ab_data.csv')
print('-----INFO-----')
print(df.info())
print('-----NUNIQUE-----')
print(df.nunique())
print('-----DESCRIBE-----')
print(df.describe())
print('-----HEAD-----')
df.head()
```

```
-----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
None
-----NUNIQUE-----
user_id      290584
timestamp    294478
group         2
landing_page  2
converted     2
dtype: int64
-----DESCRIBE-----
           user_id      converted
count  294478.000000  294478.000000
mean    787974.124733      0.119659
std      91210.823776      0.324563
min      630000.000000      0.000000
25%      709032.250000      0.000000
50%      787933.500000      0.000000
75%      866911.750000      0.000000
max      945999.000000      1.000000
-----HEAD-----
```

```
[3]:   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.

```
[4]: print('Number of lines:', len(df))
```

Number of lines: 294478

c. The number of unique users in the dataset.

```
[5]: unique_usr = df.user_id.nunique()
print('number of unique users id: ', unique_usr)
```

number of unique users id: 290584

d. The proportion of users converted.

```
[6]: usr_conv_prop = df.drop_duplicates(subset='user_id').converted.sum() / df.
    ↪ drop_duplicates(subset='user_id').converted.count()

usr_conv_prop
```

```
[6]: 0.1195695564793657
```

e. The number of times the new\_page and treatment don't line up.

```
[7]: mismatch_cols = df.query('(group == "treatment" & landing_page != "new_page") |
    ↪ (group != "treatment" & landing_page == "new_page")')
print("-----")
print("number of mismatched lines: " , len(mismatch_cols))
print("-----")
```

-----  
number of mismatched lines: 3893  
-----

f. Do any of the rows have missing values?

**No missing values**

```
[8]: 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** 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]: df2 = df
df2 = df2.drop(df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].index)
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290585 entries, 0 to 294477
Data columns (total 5 columns):
user_id          290585 non-null int64
timestamp        290585 non-null object
group            290585 non-null object
landing_page     290585 non-null object
converted        290585 non-null int64
dtypes: int64(2), object(3)
memory usage: 13.3+ MB
```

```
[10]: # 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]
```

```
[10]: 0
```

3. Use **df2** and the cells below to answer questions for **Quiz3** in the classroom.

- a. How many unique **user\_ids** are in **df2**?

```
[11]: print("Number of unique user_ids in df2", df2.user_id.nunique())
```

```
Number of unique user_ids in df2 290584
```

- b. There is one **user\_id** repeated in **df2**. What is it?

\$Reference: \$ <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.duplicated.html>

```
[12]: 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
```

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

```
[13]: df2[df2.user_id.duplicated(keep=False)]
```

```
[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`.

```
[14]: df2.drop_duplicates(subset = 'user_id', keep='first', inplace=True)
```

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?

```
[72]: print("probability of an individual converting:", df2.converted.mean())
```

probability of an individual converting: 0.11959708724499628

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

```
[16]: control_mask = df2['group'] == 'control'
      pnew = df2[control_mask].converted.mean()
      pnew
```

```
[16]: 0.1203863045004612
```

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

```
[17]: treatment_mask = df2['group'] == 'treatment'
      pold = df2[treatment_mask].converted.mean()
      pold
```

```
[17]: 0.11880806551510564
```

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

```
[18]: (df2['landing_page'] == 'new_page').mean()
```

```
[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 is not sufficient evidence to say that the new page leads to more conversion rates, neither there is statistical significance to do so. The conversion rates are very close to each other.

### 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.

$$H_0 : p_{old} \leq p_{new}$$

$$H_1 : p_{old} > p_{new}$$

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]: pnew_null = usr_conv_prop
      pnew_null
```

```
[19]: 0.1195695564793657
```

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

```
[20]: pold_null = usr_conv_prop
      pold_null
```

```
[20]: 0.1195695564793657
```

```
[21]: pnew_null-pold_null
```

```
[21]: 0.0
```

c. What is  $n_{new}$ ?

```
[22]: nnew = len(df2[treatment_mask])
      nnew
```

```
[22]: 145310
```

d. What is  $n_{old}$ ?

```
[73]: nold = len(df2[control_mask])
      print ("n_ old value: ", nold)
```

```
n_ old value: 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`.

```
[76]: new_page_converted = np.random.choice([0,1],size=nnew, p_
      ↪=[1-pnew_null,pnew_null])
      pnew_sim = new_page_converted.mean()
      print ("p_new sim : ", pnew_sim)
```

```
p_new : 0.1201293785699539
```

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`.

```
[78]: old_page_converted = np.random.choice([0,1],size=nnew, p_
      ↪=[1-pold_null,pold_null])
      pold_sim = old_page_converted.mean()
      print("p_old sim : ", pold_sim)
```

```
p_old sim : 0.12070745303145
```

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

```
[79]: print("diff p_new - p_old (simulated): ",pnew_sim - pold_sim)
```

```
diff p_new - p_old (simulated): -0.0005780744614961031
```

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]: p_diffs = []
      for _ in range(10000):
          new_page_sample = np.random.choice([0,1],size=nnew, p_
          ↪=[1-pnew_null,pnew_null])
```

```

pnew_sample = new_page_sample.mean()
old_page_sample = np.random.choice([0,1],size=nnew, p_u
↪=[1-pold_null,pold_null])
pold_sample = old_page_sample.mean()
p_diffs.append(pnew_sample - pold_sample)

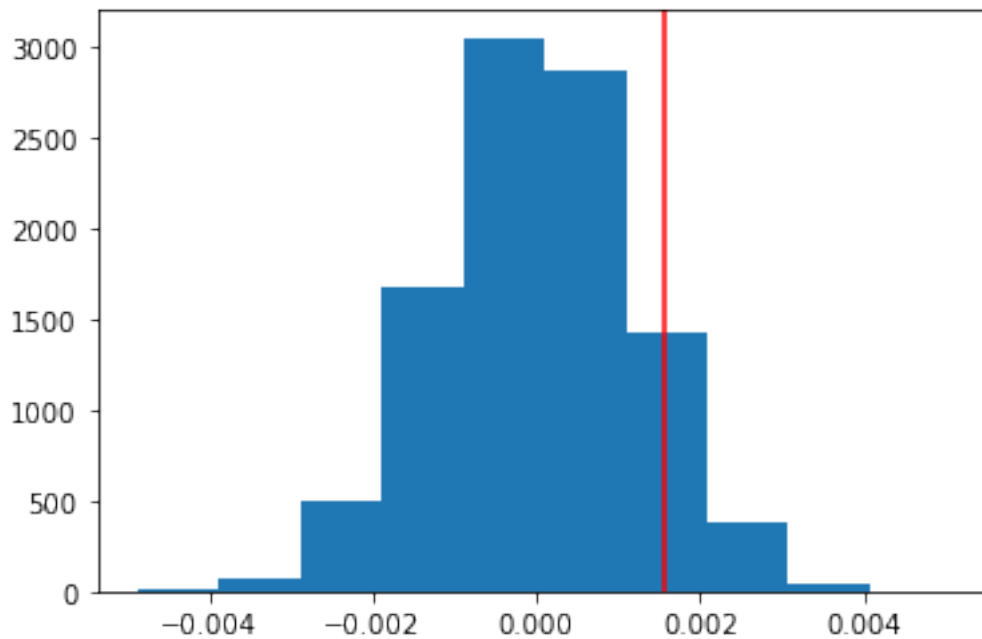
```

- 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]: plt.hist(p_diffs);
      plt.axvline(pnew-pold,color='r');

```



- j. What proportion of the **p\_diffs** are greater than the actual difference observed in **ab\_data.csv**?

```

[29]: pvalue = (p_diffs > (pnew - pold)).mean()
      pvalue

```

[29]: 0.1008

- 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?

**Answer:** In the *j* part, the “*p-value*” of the difference between the conversion ratios of the control and the treatment groups was calculated based in the null distribution. In this case the null hypothesis is that there is no difference between the conversions ration for the control and the treatment groups.



*The pvalue is the probability to observe the difference values above the observed in the provided data assuming that the null hypothesis is true*

- 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 to the number of rows associated with the old page and new pages, respectively.

```
[30]: from statsmodels.stats.proportion import proportions_ztest
```

```
convert_old = df2[control_mask].converted.sum()
convert_new = df2[treatment_mask].converted.sum()
n_old = nold
n_new = nnew

(convert_new,n_new,convert_old,n_old)
```

```
[30]: (17264, 145310, 17489, 145274)
```

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

**Z-test from stats model is two-sided by default so the alternative for one-side need to be used**

reference: [https://www.statsmodels.org/stable/generated/statsmodels.stats.proportion.proportions\\_ztest.html](https://www.statsmodels.org/stable/generated/statsmodels.stats.proportion.proportions_ztest.html)

```
[31]: stat, pval = proportions_ztest([convert_old,convert_new], \
                                   [n_old,n_new],alternative = 'larger')
print ('Z-test statistic:', stat)
print('pval for the test: ',pval)
```

```
Z-test statistic: 1.3109241984234394
```

```
pval for the test: 0.09494168724097551
```

- 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.**?

*Yes they agree, the z-test statistic means the distance from the mean, scaled by the standard deviation. The p-value has same mean.*

### Part III - A regression approach

1. In this final part, you will see that the result you achieved in the previous A/B test 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.**

- 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**.

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

df_new = df
df_new['intercept'] = 1
df_new['ab_page'] = (df_new.group == 'control').astype(int)
```

- 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.

```
[33]: df_new.head()
```

```
[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	

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

```
[34]: X = df_new[['intercept', 'ab_page']]

y = df_new.converted

model = sm.Logit(y,X)
result = model.fit()
```

```
Optimization terminated successfully.
Current function value: 0.366243
Iterations 6
```

```
[ ]:
```

- d. Provide the summary of your model below, and use it as necessary to answer the following questions.

```
[37]: result.summary2()
```

```
[37]: <class 'statsmodels.iolib.summary2.Summary'>
      """
                Results: Logit
=====
Model:                Logit                Pseudo R-squared: 0.000
Dependent Variable: converted                AIC:                215704.9004
Date:                2021-11-01 14:51        BIC:                215726.0864
No. Observations:    294478                Log-Likelihood:    -1.0785e+05
Df Model:            1                    LL-Null:            -1.0785e+05
Df Residuals:        294476                LLR p-value:        0.21611
Converged:            1.0000                Scale:            1.0000
No. Iterations:      6.0000
=====
                Coef.    Std.Err.    z      P>|z|    [0.025    0.975]
-----
intercept    -2.0027    0.0081  -248.7807  0.0000   -2.0185   -1.9869
ab_page       0.0140    0.0114    1.2369  0.2161   -0.0082    0.0363
=====
      """
```

- 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 pvalue for the **ab\_page** is equal to 0.216. It differs from the **Part II** because for the logistic regression model the null hypotessis is defined as the coefficient is 0 and the alternative hypothesis says that the coefficient is not equal to 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?

More features can better explain the conversion rate, but as all multilinear regression problem some other problems may arise. Like Multicollinearity, Outliers/Leverage Points, etc..

- 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 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.

Looking to the following summaries it looks that there is a weak dependence of conversion rate on user's Country, p values for countries dummy variables is quite high, the stronger significance are users from Canada (CA). It possible to investigate dependence to some Time based characteristic, looking at week working or day of the week. But no strong significance found.

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

Converting timestamp column to datetime.

```
[39]: df_new['timestamp'] = pd.to_datetime(df_new.timestamp)
```

Extracting time based features for future use.

```
[40]: df_new['dayofweek'] = df_new.timestamp.dt.day_name()
df_new['weekday'] = (df_new.timestamp.dt.dayofweek < 4).astype(int)
df_new = df_new.join(pd.get_dummies(df_new.dayofweek))
```

```
[41]: df_new.head()
```

```
[41]:
```

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

	dayofweek	weekday	Friday	Monday	Saturday	Sunday	Thursday	\
user_id								
834778	Saturday	0	0	0	1	0	0	
928468	Monday	1	0	1	0	0	0	
822059	Monday	1	0	1	0	0	0	
711597	Sunday	0	0	0	0	1	0	
710616	Monday	1	0	1	0	0	0	

	Tuesday	Wednesday
user_id		
834778	0	0
928468	0	0
822059	0	0
711597	0	0
710616	0	0

```
[42]: ### Create the necessary dummy variables
df_new = df_new.join(pd.get_dummies(df_new.country))
df_new['ab_page'] = (df_new.group == 'control').astype(int)
df_new['intercept'] = 1
df_new.head()
```

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

```
      dayofweek  weekday  Friday  Monday  Saturday  Sunday  Thursday  \
user_id
834778      Saturday          0          0          0          1          0          0
928468      Monday           1          0          1          0          0          0
822059      Monday           1          0          1          0          0          0
711597      Sunday           0          0          0          0          1          0
710616      Monday           1          0          1          0          0          0
```

```
      Tuesday  Wednesday  CA  UK  US  ab_page  intercept
user_id
834778          0          0  0  1  0          1          1
928468          0          0  0  0  1          0          1
822059          0          0  0  1  0          0          1
711597          0          0  0  1  0          1          1
710616          0          0  0  1  0          0          1
```

```
[50]: X = df_new[['intercept', 'ab_page', 'CA', 'UK']]
      y = df_new['converted']

      model = sm.Logit(y,X)
      result = model.fit()

      result.summary2()
```

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

```
[50]: <class 'statsmodels.iolib.summary2.Summary'>
      """
```

```

                                Results: Logit
=====
Model:                Logit                Pseudo R-squared: 0.000
Dependent Variable:    converted              AIC:                212781.1253
Date:                 2021-11-01 14:53        BIC:                212823.4439
No. Observations:     290584                 Log-Likelihood:        -1.0639e+05
Df Model:              3                     LL-Null:              -1.0639e+05
Df Residuals:          290580                 LLR p-value:           0.17599
Converged:             1.0000                 Scale:                1.0000
```

No. Iterations: 6.0000

	Coef.	Std.Err.	z	P> z	[0.025	0.975]
intercept	-2.0042	0.0089	-224.5600	0.0000	-2.0217	-1.9867
ab_page	0.0149	0.0114	1.3069	0.1912	-0.0075	0.0374
CA	-0.0408	0.0269	-1.5161	0.1295	-0.0934	0.0119
UK	0.0099	0.0133	0.7433	0.4573	-0.0162	0.0359

"""

- 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.

```
[43]: ### Fit Your Linear Model And Obtain the Results
```

```
[ ]: df_new['ab_country'] = dff_new()
```

```
[ ]:
```

```
[44]: X = df_new[['intercept', 'ab_page', 'CA', 'UK']]
      y = df_new['converted']

      model = sm.Logit(y,X)
      result = model.fit()
```

Optimization terminated successfully.  
Current function value: 0.366113  
Iterations 6

```
[45]: result.summary2()
```

```
[45]: <class 'statsmodels.iolib.summary2.Summary'>
      """
```

```

                        Results: Logit
=====
Model:                  Logit                  Pseudo R-squared: 0.000
Dependent Variable:    converted                AIC:                212781.1253
Date:                  2021-11-01 14:51         BIC:                212823.4439
No. Observations:      290584                  Log-Likelihood:        -1.0639e+05
Df Model:               3                      LL-Null:              -1.0639e+05
Df Residuals:           290580                  LLR p-value:           0.17599
Converged:              1.0000                  Scale:                1.0000
No. Iterations:         6.0000
```

	Coef.	Std.Err.	z	P> z	[0.025	0.975]
intercept	-2.0042	0.0089	-224.5600	0.0000	-2.0217	-1.9867
ab_page	0.0149	0.0114	1.3069	0.1912	-0.0075	0.0374
CA	-0.0408	0.0269	-1.5161	0.1295	-0.0934	0.0119
UK	0.0099	0.0133	0.7433	0.4573	-0.0162	0.0359

"""

[46]: *### Fit Your Linear Model And Obtain the Results*

```
X = df_new[['intercept', 'Monday', 'Tuesday', 'Wednesday',
            ↪ 'Thursday', 'Friday', 'Saturday']]
y = df_new['converted']

model = sm.Logit(y,X)
result = model.fit()
```

Optimization terminated successfully.

Current function value: 0.366112

Iterations 6

[47]: result.summary2()

[47]: <class 'statsmodels.iolib.summary2.Summary'>

"""

Results: Logit

```
=====
Model:                Logit                Pseudo R-squared: 0.000
Dependent Variable:    converted            AIC:                212786.5352
Date:                 2021-11-01 14:51      BIC:                212860.5927
No. Observations:     290584              Log-Likelihood:      -1.0639e+05
Df Model:              6                  LL-Null:             -1.0639e+05
Df Residuals:          290577             LLR p-value:         0.47741
Converged:             1.0000             Scale:               1.0000
No. Iterations:        6.0000
```

	Coef.	Std.Err.	z	P> z	[0.025	0.975]
intercept	-2.0069	0.0155	-129.5790	0.0000	-2.0373	-1.9766
Monday	0.0253	0.0211	1.1990	0.2305	-0.0161	0.0668
Tuesday	0.0101	0.0210	0.4820	0.6298	-0.0311	0.0513
Wednesday	0.0177	0.0219	0.8072	0.4196	-0.0252	0.0605
Thursday	0.0141	0.0219	0.6443	0.5194	-0.0289	0.0571
Friday	-0.0172	0.0220	-0.7817	0.4344	-0.0604	0.0259

```
Saturday      0.0223    0.0218    1.0205  0.3075  -0.0205    0.0651
=====
```

```
"""
```

Using time based dummy variables

```
[48]: X = df_new[['intercept','weekday','Monday', 'Tuesday', 'Wednesday',
↳ 'Thursday', 'Friday', 'Saturday']]
y = df_new['converted']
model = sm.Logit(y,X)
result = model.fit()
```

Optimization terminated successfully.

Current function value: 0.366112

Iterations 8

```
[49]: result.summary2()
```

```
[49]: <class 'statsmodels.iolib.summary2.Summary'>
"""
```

Results: Logit

```
=====
Model:                Logit                Pseudo R-squared:  0.000
Dependent Variable:    converted            AIC:                212786.5352
Date:                  2021-11-01 14:51      BIC:                212860.5927
No. Observations:     290584                Log-Likelihood:     -1.0639e+05
Df Model:              6                    LL-Null:           -1.0639e+05
Df Residuals:         290577                LLR p-value:        0.47741
Converged:             1.0000                Scale:             1.0000
No. Iterations:       8.0000
```

```
-----
              Coef.    Std.Err.    z    P>|z|    [0.025    0.975]
-----
intercept -2.0069      0.0155 -129.5790 0.0000    -2.0373    -1.9766
weekday   0.0135  331980.6498    0.0000 1.0000   -650670.1038  650670.1307
Monday    0.0119  331980.6498    0.0000 1.0000   -650670.1053  650670.1291
Tuesday   -0.0033  331980.6498   -0.0000 1.0000   -650670.1205  650670.1139
Wednesday 0.0042  331980.6498    0.0000 1.0000   -650670.1130  650670.1214
Thursday  0.0007  331980.6498    0.0000 1.0000   -650670.1165  650670.1179
Friday    -0.0172     0.0220  -0.7817 0.4344    -0.0604     0.0259
Saturday  0.0223     0.0218    1.0205 0.3075    -0.0205     0.0651
=====
```

```
"""
```

```
[68]: from statsmodels.formula.api import logit
```



Using 'R' like formulas to set interactions: <https://www.statsmodels.org/stable/examples/notebooks/generated/discr>

```
[69]: model = logit("converted ~ ab_page* ( UK + CA )", data=df_new).fit()
```

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

```
[70]: model.summary2()
```

```
[70]: <class 'statsmodels.iolib.summary2.Summary'>
      """
```

```

                        Results: Logit
=====
Model:                  Logit                  Pseudo R-squared: 0.000
Dependent Variable: converted                  AIC:                212782.6602
Date:                  2021-11-01 15:35        BIC:                212846.1381
No. Observations:      290584                  Log-Likelihood:      -1.0639e+05
Df Model:              5                      LL-Null:            -1.0639e+05
Df Residuals:          290578                  LLR p-value:         0.19199
Converged:             1.0000                  Scale:             1.0000
No. Iterations:        6.0000

-----
              Coef.   Std.Err.    z      P>|z|    [0.025   0.975]
-----
Intercept    -2.0070    0.0097  -207.0454  0.0000   -2.0260   -1.9880
ab_page       0.0206    0.0137   1.5052   0.1323   -0.0062    0.0473
UK            0.0257    0.0188   1.3634   0.1728   -0.0112    0.0625
CA           -0.0644    0.0384  -1.6788   0.0932   -0.1396    0.0108
ab_page:UK   -0.0314    0.0266  -1.1807   0.2377   -0.0835    0.0207
ab_page:CA    0.0469    0.0538   0.8718   0.3833   -0.0585    0.1523
=====

      """
```

In this last part of the project it is possible to observe a small increase in statistical significance (lower p values) for predicting the conversion rate when taking in consideration the interaction between country origin and landing page.

## Conclusions

Congratulations on completing the project!

### 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](mailto: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.

[ ]:

[ ]: