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!

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.

```
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:

```
1 df=pd.read_csv('ab_data.csv')
2 df.head()
```

	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

```
1 df.tail()
```

	user_id	timestamp	group	landing_page	converted
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

```
1 df.count()
```

```
user_id         294478
timestamp       294478
group           294478
landing_page    294478
converted       294478
dtype: int64
```

b. Use the below cell to find the number of rows in the dataset.

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype
 1   timestamp     294478 non-null  object
```

c. The number of unique users in the dataset.

```
1 df['user_id'].nunique()
```

```
290584
memory usage: 11.2+ MB
```

d. The proportion of users converted.

```
1 (len(df[df['converted']==1]),df.shape[0])
```

```
(35237, 294478)
```

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

```
1 df[((df['group'] == 'treatment') == (df['landing_page'] == 'new_page')) == False].shape[0]
```

```
3893
```

f. Do any of the rows have missing values?

```
1 df.isna().any().sum()
```

```
0
```

There is no missing values in the rows

`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**.

```
1 df.drop(df[((df['group'] == 'treatment') == (df['landing_page'] == 'new_page')) == False].
```

```
1 #confirming the dropped rows
2 df.shape
```

```
(290585, 5)
```

```
#creating df2 as a copy of df
df2 = df
df2
```

	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 × 5 columns

```
#preview the top of df2
df2.head()
```

	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

```
#checking the shape of df2
df2.shape
```

```
(290585, 5)
```

```
#checking the shape of df2
df2.dtypes
```

```
user_id          int64
timestamp       object
group           object
landing_page    object
converted        int64
dtype: object
```

```
#basic statistics of df2
df2.describe()
```

	user_id	converted
count	290585.000000	290585.000000
mean	788004.825246	0.119597
std	91224.582639	0.324490
min	630000.000000	0.000000
25%	709035.000000	0.000000
50%	787995.000000	0.000000
75%	866956.000000	0.000000
max	945999.000000	1.000000

```
#checking the number of unique values
df2.nunique()
```

```
user_id         290584
timestamp       290585
group                2
landing_page         2
converted            2
dtype: int64
```

`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**?

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

```
290584
```

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

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

```
1
```

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

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

```
      user_id                   timestamp      group landing_page  converted
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**.

```
df2['user_id'].drop_duplicates(inplace=True)
```

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

```
False
```

`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?

```
#Probability of an individual converting regardless of the page they receive
df2['converted'].mean()
```

```
0.11959667567149027
```

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

```
# probability of  an individual in the control group converting
df2.query('group == "control"')['converted'].mean()
```

```
0.1203863045004612
```

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

```
#probabilityan of an individual in the treatment group converted
df2.query('group == "treatment"')['converted'].mean()
```

```
0.11880724790277405
```

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

```
#Probability that an individual received the new page
```

```
#Probability that an individual received the new page
((df2[df2['landing_page'] == "new_page"]).count()['landing_page'])/df2.shape[0]
```

```
0.5000636646764286
```

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.

Probability of an individual converting regardless of the page they receive is 0.11959667567149027, the probability of an individual in the control group converting is 0.1203863045004612, probabilityan of an individual in the treatment group converting is 0.11880724790277405. It is evident that there is no sufficient evidence that the new treatment page leads to more conversions because the probabilities are almost equal

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.

$$H_0 : p_{new} - p_{old} \geq 0$$
$$H_1 : p_{new} - p_{old} < 0$$

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

```
# convert rate for  pnew
p_new_convert_rate = df2['converted'].mean()
p_new_convert_rate
```

```
0.11959667567149027
```

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

```
# convert rate for  pold
p_old_convert_rate = df2['converted'].mean()
p_old_convert_rate
```

```
0.11959667567149027
```

c. What is n_{new}?

```
#querry df2 where group  is treatment
number_new = df2.query('group == "treatment"').shape[0]
number_new
```

```
145311
```

d. What is n_{old}?

```
#querry df2 where group  is control
number_old = df2.query('group == "control"').shape[0]
number_old
```

```
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**.

```
new_page_converted = np.random.choice([0,1], size=number_new, p=[1-p_new_convert_rate, p_r
new_page_converted
```

```
array([0, 0, 1, ..., 1, 0, 0])
```

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**.

```
old_page_converted=np.random.choice([0,1], size=number_old, p=[1-p_old_convert_rate, p_old
old_page_converted
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

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

```
diff=new_page_converted.mean() - old_page_converted.mean()
diff
```

```
0.0005407700404348043
```

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**.

```
p_diffs = []
for i in range(10000):
    new_page_converted = np.random.choice([0,1], size=number_new, p=[1-p_new_convert_rate,
    old_page_converted=np.random.choice([0,1], size=number_old, p=[1-p_old_convert_rate, p
    p_diffs.append(new_page_converted.mean() - old_page_converted.mean())
p_diffs = np.array(p_diffs)
```

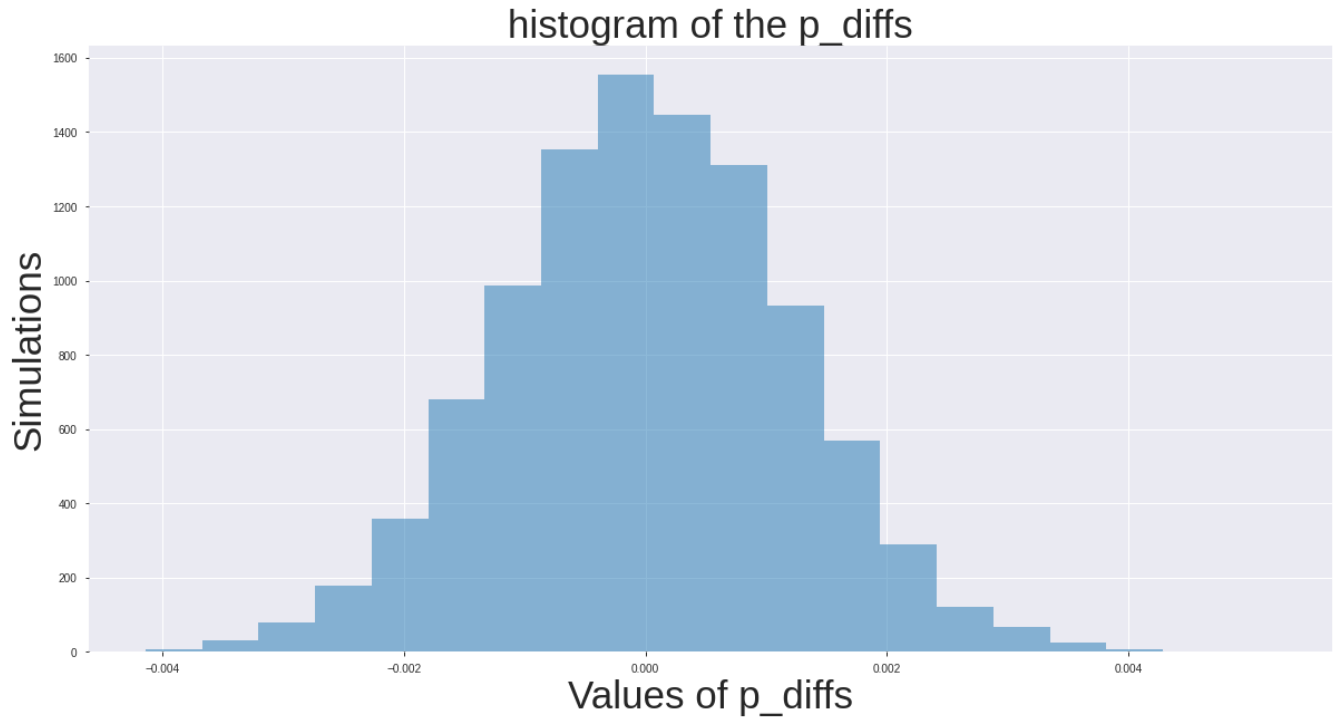
i. Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

```
#conda install -f matplotlib
```

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.figure(figsize=(20,10))
```

```
plt.hist(p_diffs, alpha = 0.5,bins=20);
plt.title("histogram of the p_diffs",fontsize=35)
plt.xlabel("Values of p_diffs",fontsize=35)
plt.ylabel("Simulations",fontsize=35)
```

```
Text(0, 0.5, 'Simulations')
```



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

```
# Calculating the actual difference observed in df2
observed_diffs = df2.query('group=="treatment"').converted.mean() - df2.query('group=="cor
observed_diffs
```

```
-0.0015790565976871451
```

```
(p_diffs > observed_diffs).mean()
```

```
(p_diffs > observed_diffs).mean()
```

```
0.9073
```

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?

This is the p_value, it indicates whether to reject the Null hypothesis, it also determines the error Type.

When it is bellow 0.5, the Null hypothesis is accepted, in this case it is ~0.9 hence the Null hypothesis is rejected.

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.

```
import statsmodels.api as sm
# find the conversions per page
convert_old = sum((df2.group=='control')&(df2.converted==1))
convert_new = sum((df2.group=='treatment')&(df2.converted==1))
# find the number of samples that received each page
number_old = df2.query('group == "control"').shape[0]
number_new = df2.query('group == "treatment"').shape[0]
```

```
# or we can use the implementation from statsmodels
# where we pass in the success (they call the argument counts)
# and the total number for each group (they call the argument nobs,
# number of observations)
counts = [convert_new, convert_old]
nobs = [number_new, number_old]
z_score, p_value = sm.stats.proportions_ztest(counts, nobs, alternative = 'larger')
z_score, p_value
```

```
(-1.3116075339133115, 0.905173705140591)
```

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

```
zscore, pvalue = sm.stats.proportions_ztest(counts, nobs, alternative = 'two-sided')
print('zscore = {:.3f}, pvalue = {:.3f}'.format(zscore, pvalue))
```

```
zscore = -1.312, pvalue = 0.190
```

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 zscore = -1.312,while the pvalue = 0.190 which does not agree with the findings in parts j. and k. This indicates that the old and new pages have no effect to the conversion rate.

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?

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**.

```
#loading df and preview the top
df = pd.read_csv('ab_data.csv')
df.head()
```

	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

```
#creating a copy of df2
df2=df.copy()
```

```
#group  landing_page
df2[['control','treatment']]=pd.get_dummies(df2['group'])
df2[['new_page','old_page']]=pd.get_dummies(df2['landing_page'])
df2=df2.drop('group',axis=1)
df2=df2.drop('landing_page',axis=1)
df2=df2.drop('treatment',axis=1)
df2=df2.drop('old_page',axis=1)
```

```
# preview the top of df2
df2.head()
```

	user_id	timestamp	converted	control	new_page
0	851104	2017-01-21 22:11:48.556739	0	1	0
1	804228	2017-01-12 08:01:45.159739	0	1	0
2	661590	2017-01-11 16:55:06.154213	0	0	1
3	853541	2017-01-08 18:28:03.143765	0	0	1
4	864975	2017-01-21 01:52:26.210827	1	1	0
...	...	...	...	...	...
294473	751197	2017-01-03 22:28:38.630509	0	1	0
294474	945152	2017-01-12 00:51:57.078372	0	1	0
294475	734608	2017-01-22 11:45:03.439544	0	1	0
294476	697314	2017-01-15 01:20:28.957438	0	1	0
294477	715931	2017-01-16 12:40:24.467417	0	0	1

294478 rows × 5 columns

```
#reanming columns control to ab_page and landing to new_page
df2['ab_page']=df2['control']
df2['landing_page']=df2['new_page']
```

```
#dropping columns that are not needed after feature engineering
df2=df2.drop('control',axis=1)
df2=df2.drop('new_page',axis=1)
```

```
#preview top of df2
df2.head()
```

	user_id	timestamp	converted	ab_page	landing_page
0	851104	2017-01-21 22:11:48.556739	0	1	0
1	804228	2017-01-12 08:01:45.159739	0	1	0
2	661590	2017-01-11 16:55:06.154213	0	0	1

```
#loop to change date columns to date type
time_cols = ['timestamp']
for i in range(1):
    new =pd.to_datetime(df2[time_cols[i]],format='%Y-%m-%d %H:%M:%S').dt.tz_localize(None)
    df2[time_cols[i]] = new
df2.head()
```

	user_id	timestamp	converted	ab_page	landing_page
0	851104	2017-01-21 22:11:48.556739	0	1	0
1	804228	2017-01-12 08:01:45.159739	0	1	0
2	661590	2017-01-11 16:55:06.154213	0	0	1
3	853541	2017-01-08 18:28:03.143765	0	0	1
4	864975	2017-01-21 01:52:26.210827	1	1	0

```
#extracting the day,month and year into columns
#This is to enable analysis per day,week,month and year
df2['day'] = df2['timestamp'].dt.day
df2['week'] = df2['timestamp'].dt.week
df2['month'] = df2['timestamp'].dt.month
df2['year'] = df2['timestamp'].dt.year
df2['hour'] = df2['timestamp'].dt.hour
df2['minute'] = df2['timestamp'].dt.minute
df2
```

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**?

p-value associated with ab_page=0.682

The null and alternative hypotheses associated with your regression model

$$H_0 : \beta_1 = 0$$
$$H_1 : \beta_1 \neq 0$$

p-value associated with ab_page=0.682, it differs because the model performs a 2-sided Test,In this case it indicates that there is no relation between a user seeing either page(old/new) and the conversion .This because it is more than the recomended value of p_value(0.05). As a result the experiment has a Type I error and the Null hypothesis is rejected.

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 a good practice as a Data analyst to be inquistive in suspect all the available variables that influences a given result

The only disadvantages of adding additional terms is the human effort in terms of feature engineering and increased computational cost

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.

```
#create the dataset and preview the top
df_new=pd.read_csv('countries.csv')
df_new.head()
```

	user_id	timestamp	converted	ab_page	landing_page	day	week	month	ye
0	851104	2017-01-21 22:11:48.556739	0	1	0	21	3	1	20
1	804228	2017-01-12 08:01:45.159739	0	1	0	12	2	1	20

```
#dropping columns that are not needed after feature engineering
df2=df2.drop(['user_id','timestamp'],axis=1)
df2
```

	converted	ab_page	landing_page	day	week	month	year	hour	minute
0	0	1	0	21	3	1	2017	22	11
1	0	1	0	12	2	1	2017	8	1
2	0	0	1	11	2	1	2017	16	55
3	0	0	1	8	1	1	2017	18	28
4	1	1	0	21	3	1	2017	1	52
...	...	...	...	...	...	...	...	...	...
294473	0	1	0	3	1	1	2017	22	28
294474	0	1	0	12	2	1	2017	0	51
294475	0	1	0	22	3	1	2017	11	45
294476	0	1	0	15	2	1	2017	1	20
294477	0	0	1	16	3	1	2017	12	40

294478 rows × 9 columns

```
df2.columns
```

```
Index(['converted', 'ab_page', 'landing_page', 'day', 'week', 'month', 'year',
       'hour', 'minute'],
      dtype='object')
```

```

# manually add the intercept
# Adding an intercept column
df2['intercept'] = 1.0  # so we don't need to use sm.add_constant every time
```

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.

```
import statsmodels as sm
import statsmodels.regression.linear_model as sm
```

```
import statsmodels.regression.linear_model as sm
import statsmodels.api as sm
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
# Instantiating the regression model
logit_mod = sm.Logit(df2['converted'], df2[['ab_page','landing_page','day','week','month',
# Fitting the model
results = logit_mod.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.366224
         Iterations 6
```

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

```
#get model summary
results.summary()
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:1286: RuntimeWarning:
  bse_ = np.sqrt(np.diag(self.cov_params()))
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:903: Runtim
  return (a < x) & (x < b)
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:903: Runtim
  return (a < x) & (x < b)
/usr/local/lib/python3.6/dist-packages/scipy/stats/_distn_infrastructure.py:1912: Runti
  cond2 = cond0 & (x <= _a)
```

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	294478
Model:	Logit	**Df Residuals:**	294471
Method:	MLE	**Df Model:**	6
Date:	Wed, 24 Jun 2020	**Pseudo R-squ.:**	5.839e-05
Time:	07:44:24	**Log-Likelihood:**	-1.0784e+05
converged:	True	**LL-Null:**	-1.0785e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.04995

	coef	std err	z	P>|z|	[0.025	0.975]
ab_page	-0.0204	0.050	-0.410	0.682	-0.118	0.077
landing_page	-0.0353	0.050	-0.710	0.478	-0.133	0.062
day	-0.0018	0.003	-0.629	0.530	-0.007	0.004
week	0.0233	0.020	1.183	0.237	-0.015	0.062
month	2.475e-06	nan	nan	nan	nan	nan
year	-0.0010	nan	nan	nan	nan	nan
hour	0.0021	0.001	2.587	0.010	0.001	0.004
minute	-0.0002	0.000	-0.631	0.528	-0.001	0.000

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in **Part II**?

	user_id	country
0	834778	UK
1	928468	US

```
#get unique values
df_new['country'].nunique()
```

```
3
```

```
### Create the necessary dummy variables
df_new[['CA','US','UK']]=pd.get_dummies(df_new['country'])
df_new
```

	user_id	country	CA	US	UK
0	834778	UK	0	1	0
1	928468	US	0	0	1
2	822059	UK	0	1	0
3	711597	UK	0	1	0
4	710616	UK	0	1	0
...	...	...	...	...	...
290579	653118	US	0	0	1
290580	878226	UK	0	1	0
290581	799368	UK	0	1	0
290582	655535	CA	1	0	0
290583	934996	UK	0	1	0

290584 rows × 5 columns

```
#concatenating the columns
df2 = pd.concat([df2,df_new],axis=1)
#previewing the head
df2.head()
```

converted	ab_page	landing_page	day	week	month	year	hour	minute	intercept	u

```
#checking missing values after concatenation
df2.isna().any().sum()
```

```
5
```

```
#dropping missing values after concatenation
df2.dropna(inplace=True)
```

```
#confirming missing values after concatenation
df2.isna().any().sum()
```

```
0
```

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.

```
# manually add the intercept
# Adding an intercept column
df2['intercept'] = 1.0  # so we don't need to use sm.add_constant every time
```

```
import statsmodels as sm
import statsmodels.regression.linear_model as sm
import statsmodels.api as sm
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
# Instantiating the regression model
logit_mod = sm.Logit(df2['landing_page'], df2[['CA','US','UK']])
# Fitting the model
results = logit_mod.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.861230
         Iterations 3
```

```
results.summary()
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:492: HessianInversionW
  'available', HessianInversionWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:492: HessianInversionW
  'available', HessianInversionWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/discrete/discrete_model.py:3390: Run
  return 1 - self.llf/self.llnull
```

Logit Regression Results

Dep. Variable:	landing_page	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290581
Method:	MLE	**Df Model:**	2
Date:	Wed, 24 Jun 2020	**Pseudo R-squ.:**	inf
Time:	07:44:30	**Log-Likelihood:**	-2.5026e+05

The summary results shows all the p_values associated with the Countries as

CA p_value(0.601)

US p_value(0.114)

UK p_value(0.480)

Similarly this indicates that they all have no statistical significance. The country of origin has no effect on the conversion of the user.