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 (https://review.udacity.com/#!/projects/37e27304-ad47-4eb0-a1ab-8c12f60e43d0/rubric).

Part I - Probability

To get started, let's import our libraries.

In [1]:

```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
%matplotlib inline
# We are setting the seed to assure you get the same answers
# on quizzes as we set up.
random.seed(42)
```

1. Now, read in the `ab_data.csv` data. Store it in `df`. **Use your dataframe to answer the questions in Quiz 1 of the classroom.**

a. Read in the dataset and take a look at the top few rows here:

In [2]:

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

Out[2]:

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

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

In [3]:

```
row_count = df.shape[0]
print('row count = ' + str(row_count))
```

```
row count = 294478
```

c. The number of unique users in the dataset.

In [4]:

```
unique_users = df.user_id.nunique()
print('unique user count = ' + str(unique_users))
```

```
unique user count = 290584
```

d. The proportion of users converted.

In [5]:

```
unique_converted_users = df[df.converted == 1].user_id.nunique()
print('unique converted user count = ' \
      + str(unique_converted_users))
print('proportion of unique users that are converted = ' \
      + str(unique_converted_users/unique_users))
```

```
unique converted user count = 35173
proportion of unique users that are converted = 0.12104245244060237
```

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

In [6]:

```
df.groupby(['group', 'landing_page']).count()
```

Out[6]:

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

In [7]:

```
treatment_old_page = \
    df[(df.group == 'treatment') & \
       (df.landing_page == 'old_page')].shape[0]
control_new_page = \
    df[(df.group == 'control') & \
       (df.landing_page == 'new_page')].shape[0]

print('# times treatment group receives old_page: ' \
    + str(treatment_old_page))
print('  # times control group receives new_page: ' \
    + str(control_new_page))
print('                                Sum of both: ' \
    + str(treatment_old_page + control_new_page))
```

```
# times treatment group receives old_page: 1965
  # times control group receives new_page: 1928
                               Sum of both: 3893
```

f. Do any of the rows have missing values?

In [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
```

In [9]:

```
print(df.isnull().sum())
print('\nTherefore, no.')
```

```
user_id         0
timestamp       0
group           0
landing_page    0
converted       0
dtype: int64

Therefore, no.
```

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**.

In [10]:

```
drop_index = df[( (df.group == 'treatment') & \
                  (df.landing_page == 'old_page') ) | \
                ( (df.group == 'control') & \
                  (df.landing_page == 'new_page') ) ].index

df2 = df.drop(drop_index)
```

In [11]:

```
# 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]
```

Out[11]:

```
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 [12]:

```
df2_unique_users = df2.user_id.nunique()
print('df2 unique user count = ' + str(df2_unique_users))
```

```
df2 unique user count = 290584
```

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

In [13]:

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

repeat_user_id = str(df2[df2.duplicated('user_id')].user_id)[8:14]
```

```
2893    773192
Name: user_id, dtype: int64
```

In [14]:

```
print('repeat user id = ' + repeat_user_id)
```

```
repeat user id = 773192
```

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

In [15]:

```
df2[df2.user_id == int(repeat_user_id)]
```

Out[15]:

	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 [16]:

```
print('shape before dropping = ' + str(df2.shape))
df2.drop([1899], inplace=True)
print(' shape after dropping = ' + str(df2.shape))
```

```
shape before dropping = (290585, 5)
 shape after dropping = (290584, 5)
```

4. Use **df2** in the below cells to answer the quiz questions related to **Quiz 4** in the classroom.

a. What is the probability of an individual converting regardless of the page they receive?

In [17]:

```
df2.converted.mean()
```

Out[17]:

```
0.11959708724499628
```

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

In [18]:

```
control_conv = df2[df2['group'] == 'control'].converted.mean()
control_conv
```

Out[18]:

0.1203863045004612

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

In [19]:

```
treatment_conv = df2[df2['group'] == 'treatment'].converted.mean()
treatment_conv
```

Out[19]:

0.11880806551510564

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

In [20]:

```
new_page_count = df2[df2.landing_page == 'new_page'].user_id.count()
new_page_count / df2.shape[0]
```

Out[20]:

0.50006194422266881

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.

There does not appear to be sufficient evidence to conclude that the new treatment page produces more conversions than the current control page. The probability that a user who receives the treatment page will convert is actually slightly less than the probability that a user who receives the control page will convert.

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.

$$H_0 : p_{new} - p_{old} \leq 0$$
$$H_1 : p_{new} - p_{old} > 0$$

Or, verbally:

H_0: The likelihood of conversion for a user receiving the new page is less than or equal to the likelihood of conversion for a user receiving the old page.

H_1: The likelihood of conversion for a user receiving the new page is greater than the likelihood of conversion for a user receiving the old page.

2. Assume under the null hypothesis, p_{new} and p_{old} both have "true" success rates equal to the **converted** success rate regardless of page - that is p_{new} and p_{old} are equal. Furthermore, assume they are equal to the **converted** rate in **ab_data.csv** regardless of the page.

Use a sample size for each page equal to the ones in **ab_data.csv**.

Perform the sampling distribution for the difference in **converted** between the two pages over 10,000 iterations of calculating an estimate from the null.

Use the cells below to provide the necessary parts of this simulation. If this doesn't make complete sense right now, don't worry - you are going to work through the problems below to complete this problem. You can use **Quiz 5** in the classroom to make sure you are on the right track.

a. What is the **convert rate** for p_{new} under the null?

In [21]:

```
p_new = df2.converted.mean()
print(p_new)
```

```
0.11959708724499628
```

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

In [22]:

```
p_old = df2.converted.mean()
print(p_old)
```

```
0.11959708724499628
```

c. What is n_{new}?

In [23]:

```
n_new = df2[df2.landing_page == 'new_page'].user_id.count()
print(n_new)
```

```
145310
```

d. What is n_{old}?

In [24]:

```
n_old = df2[df2.landing_page == 'old_page'].user_id.count()
print(n_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**.

In [25]:

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

f. Simulate n_{old} transactions with a convert rate of p_{old} under the null. Store these n_{old} 1's and 0's in **old_page_converted**.

In [26]:

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

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

In [27]:

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

Out[27]:

```
-0.00094496826737285045
```

h. Simulate 10,000 p_{new} - p_{old} values using this same process similarly to the one you calculated in parts **a. through g.** above. Store all 10,000 values in a numpy array called **p_diffs**.

In [28]:

```
new_converted_simulation = \
    np.random.binomial(n_new, p_new, 10000)/n_new
old_converted_simulation = \
    np.random.binomial(n_old, p_old, 10000)/n_old
p_diffs = new_converted_simulation - old_converted_simulation
```

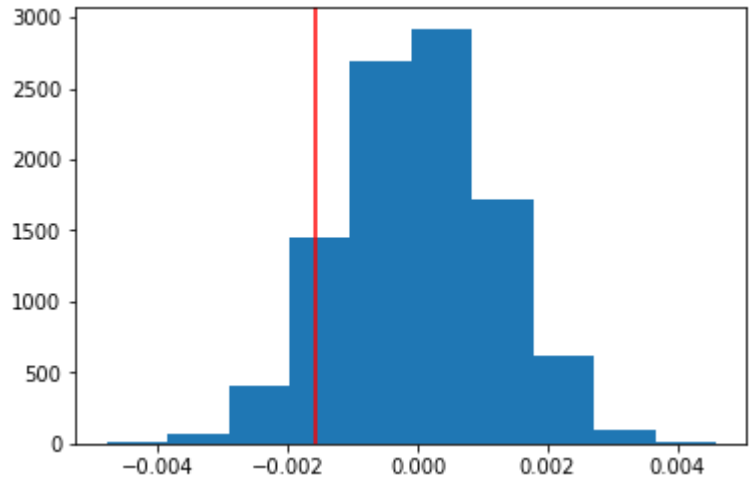
i. Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

This boils down to a computation of the "spread" of the data, assuming that the probability of converting a given user is the same whether they see the treatment page or the control page.

In [29]:

```
obs_diff = treatment_conv - control_conv

plt.hist(p_diffs);
plt.axvline(x=obs_diff, color='red');
```



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

In [30]:

```
p_diffs = np.array(p_diffs)
(p_diffs > obs_diff).mean()
```

Out[30]:

```
0.90329999999999999
```

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 histogram plotted in part i contains the sampling distribution under the null hypothesis, namely, that the conversion rate of the control group is equal to the conversion rate of the treatment group. Part j involves calculating what proportion of the conversion rate differences were greater than the actual observed difference, which was calculated from the conversion rate data. The special name given to the proportion of values in the null distribution that were greater than our observed difference is the "p-value."

A low p-value (specifically, less than our alpha of 0.05) indicates that the null hypothesis is not likely to be true. Since the p-value is very large at 90%, it is likely that our statistic is from the null, and therefore we fail to reject the null hypothesis. Ultimately, this indicates that it would be best for Audacity to keep the current page.

l. We could also use a built-in to achieve similar results. Though using the built-in might be easier to code, the above portions are a walkthrough of the ideas that are critical to correctly thinking about statistical significance. Fill in the below to calculate the number of conversions for each page, as well as the number of individuals who received each page. Let `n_old` and `n_new` refer the the number of rows associated with the old page and new pages, respectively.

In [31]:

```
import statsmodels.api as sm

convert_old = df2[df2['group'] == 'control'].converted.sum()
convert_new = df2[df2['group'] == 'treatment'].converted.sum()
n_old = df2[df2['group'] == 'control'].converted.size
n_new = df2[df2['group'] == 'treatment'].converted.size
```

```
/Users/ryanwingate/anaconda3/lib/python3.6/site-packages/statsmodel
s/compat/pandas.py:56: FutureWarning: The pandas.core.datetools modu
le is deprecated and will be removed in a future version. Please use
the pandas.tseries module instead.
  from pandas.core import datetools
```

m. Now use `stats.proportions_ztest` to compute your test statistic and p-value. Here (http://knowledgetack.com/python/statsmodels/proportions_ztest/) is a helpful link on using the built in.

In [32]:

```
from scipy.stats import norm

z_score, p_value = \
    sm.stats.proportions_ztest( [ convert_new, convert_old ], \
                                [ n_new, n_old ], \
                                alternative='larger' )

print('Z-score critical value (95% confidence) to \n' \
      + '    reject the null: ' \
      + str(norm.ppf(1-(0.05/2))))

print('z_score = ' + str(z_score))
print('p_value = ' + str(p_value))
```

```
Z-score critical value (95% confidence) to
    reject the null: 1.95996398454
z_score = -1.31092419842
p_value = 0.905058312759
```

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.**?

Since the magnitude of the z-score of 1.31 falls within the range implied by the critical value of 1.96, we fail to reject the null hypothesis. The null hypothesis is that there is no statistical difference between the conversion rates for the control and treatment groups.

Additionally, since the p_value of 0.90 (note, approximately the same value as was calculated manually) is larger than the alpha value of 0.05, we fail to reject the null hypothesis.

Thus, for both the foregoing reasons, the built-in method leads to the same conclusion as the manual method, the results of which are summarized in parts j and k, above.

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**.

In [33]:

```
df2.head()
```

Out[33]:

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

In [34]:

```
df2['intercept'] = 1

df2[['drop', 'ab_page']] = pd.get_dummies(df2['group'])
df2.drop(['drop'], axis=1, inplace=True)
df2.head()
```

Out[34]:

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

c. Use **statsmodels** to import your regression model. Instantiate the model, and fit the model using the two columns you created in part **b.** to predict whether or not an individual converts.

In [35]:

```
import statsmodels.api as sm

logit_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
```

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

In [36]:

```
results = logit_mod.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366118
         Iterations 6
```

Out[36]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290582
Method:	MLE	**Df Model:**	1
Date:	Mon, 12 Feb 2018	**Pseudo R-squ.:**	8.077e-06
Time:	13:18:43	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.1899

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
intercept	-1.9888	0.008	-246.669	0.000	-2.005	-1.973
ab_page	-0.0150	0.011	-1.311	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 in this regression model is 0.19. The p-value that was returned from the built-in ztest method was ~0.90. The p-value that I calculated manually was also ~0.90.

The null hypothesis associated with a logistic regression is that there is no relationship between the dependent and independent variables. In this case, this means there is no relationship between which page a user is shown and the conversion rate. The alternative hypothesis would therefore be that there is a relationship of some sort.

The null hypothesis from part 2 is that the likelihood of conversion for a user receiving the new page is less than or equal to the likelihood of conversion for a user receiving the old page. The alternative hypothesis from part 2 is that the likelihood of conversion for a user receiving the new page is greater than the likelihood of conversion for a user receiving the old page.

The factor that accounts for the large difference in the p-values may be that part 2 hypothesized one of the pages (specifically, the new_page the treatment group received) would lead to more conversions than the other. This is different from the hypotheses of part 3, which merely predicted a difference of some sort.

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?

Additional factors may make the model more predictive, yielding greater understanding. It may also result in business insights that would not have been evident in this simpler anlysis. For example, it would be possible to have different versions of the website for different locations. It is likely that people from different countries might have different tastes in website layout.

Possible disadvantages include increased risk of human error, especially misinterpretation, as well as possibly obscuring the message the data is really trying to tell (decreasing the so-called signal-to-noise ratio).

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 (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.join.html) 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 [37]:

```
countries_df = pd.read_csv('./countries.csv')
df_new = countries_df.set_index('user_id')\
    .join(df2.set_index('user_id'), how='inner')
```

In [38]:

```
df_new.head()
```

Out[38]:

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

In [39]:

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

In [40]:

```
logit_mod_new = sm.Logit(df_new['converted'],\
    df_new[['intercept', 'ab_page', 'US', 'UK']])
results_new = logit_mod_new.fit()
results_new.summary()
```

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

Out[40]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290580
Method:	MLE	**Df Model:**	3
Date:	Mon, 12 Feb 2018	**Pseudo R-squ.:**	2.323e-05
Time:	13:18:44	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.1760

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
intercept	-2.0300	0.027	-76.249	0.000	-2.082	-1.978
ab_page	-0.0149	0.011	-1.307	0.191	-0.037	0.007
US	0.0408	0.027	1.516	0.130	-0.012	0.093
UK	0.0506	0.028	1.784	0.074	-0.005	0.106

In [41]:

```
np.exp(0.0408), np.exp(0.0506)
```

Out[41]:

```
(1.0416437559600236, 1.0519020483004984)
```

The interpretation of the foregoing variables is counterintuitive. In this case, Canada is the baseline since it was the one out of three variables that wasn't included in the regression. We would say that US users are 1.04 times as likely (or 4% more likely) to convert as Canadians users. Similarly, we would say that UK users are 1.05 times as likely (or 5% more likely) to convert as Canadian users.

The effect is not statistically significant, given the fairly large P-values. Even if it were, it is not clear that such a small difference between the different countries would be practically significant.

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 [42]:

```
# These columns indicate that a given user received both the new page
#    and lived in the country shown.
df_new['new_CA'] = df_new['ab_page']*df_new['CA']
df_new['new_UK'] = df_new['ab_page']*df_new['UK']
df_new['new_US'] = df_new['ab_page']*df_new['US']
```

In [43]:

```
df_new.head()
```

Out[43]:

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

In [44]:

```
### Fit Your Linear Model And Obtain the Results
lin_mod = sm.OLS(df_new['converted'], \
    df_new[['intercept', 'ab_page', 'US', 'new_US', 'UK', 'new_UK']])
results = lin_mod.fit()
results.summary()
```

Out[44]:

OLS Regression Results

Dep. Variable:	converted	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	1.466
Date:	Mon, 12 Feb 2018	**Prob (F-statistic):**	0.197
Time:	13:18:44	**Log-Likelihood:**	-85265.
No. Observations:	290584	**AIC:**	1.705e+05
Df Residuals:	290578	**BIC:**	1.706e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
intercept	0.1188	0.004	31.057	0.000	0.111	0.126
ab_page	-0.0069	0.005	-1.277	0.202	-0.017	0.004
US	0.0018	0.004	0.467	0.641	-0.006	0.010
new_US	0.0047	0.006	0.845	0.398	-0.006	0.016
UK	0.0012	0.004	0.296	0.767	-0.007	0.009
new_UK	0.0080	0.006	1.360	0.174	-0.004	0.020

Omnibus:	125549.436	**Durbin-Watson:**	1.996
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	414285.945
Skew:	2.345	**Prob(JB):**	0.00
Kurtosis:	6.497	**Cond. No.**	26.1

In [45]:

```
log_mod2 = sm.Logit(df_new['converted'], \
    df_new[['intercept', 'ab_page', 'US', 'new_US', 'UK', 'new_UK']])
results_log2 = log_mod2.fit()
results_log2.summary()
```

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

Out[45]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290578
Method:	MLE	**Df Model:**	5
Date:	Mon, 12 Feb 2018	**Pseudo R-squ.:**	3.482e-05
Time:	13:18:45	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.1920

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
intercept	-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
new_US	0.0469	0.054	0.872	0.383	-0.059	0.152
UK	0.0118	0.040	0.296	0.767	-0.066	0.090
new_UK	0.0783	0.057	1.378	0.168	-0.033	0.190

The foregoing present both a linear (included to comply with the comment "#Fit your linear model") and logistic regression for the case with the interaction terms. Neither effect is statistically significant, given the high P-values shown in the results. Additionally, in the case of the linear plot, the R-squared value is zero, implying a terrible fit.