Analyze_ab_test_results_notebook

May 31, 2022

1 Analyze A/B Test Results

This project explores concepts in probability, hypothesis testing, and regression. The document is organised into the following sections:

- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**

Specific programming tasks are marked with a **ToDo** tag.

Introduction

The goal of this project is to understand the results of an A/B test run by an e-commerce website to determine if the company should: - Implement the new webpage, - Keep the old webpage, or - Perhaps run the experiment longer to make their decision.

Below is the description of the data, there are a total of 5 columns:

Data columns	Purpose	Valid values
user_id	Unique ID	Int64 values
timestamp	Time stamp when the user visited the webpage	-

Data columns	Purpose	Valid values
group	In the current A/B experiment, the users are categorized into two broad groups. The `control` group users are expected to be served with `old_page`; and `treatment` group users are matched with the `new_page`. However, **some inaccurate rows** are present in the initial data, such as a `control` group user is matched with a `new_page`.	`['control', 'treatment']`
landing_page	It denotes whether the user visited the old or new webpage.	`['old_page', 'new_page']`
converted	It denotes whether the user decided to pay for the company's product. Here, `1` means yes, the user bought the product.	`[0, 1]`

Part I - Probability

```
In [1]: # importing relevant 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)
```

1.0.1 ToDo 1.1

a. Read in the `ab_data.csv` data. Store it in `df`. Use your dataframe to answer the questions in Quiz 1 of the classroom.

```
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 cell below to find the number of rows in the dataset.

```
In [3]: len(df)
```

```
Out[3]: 294478
```

```
In [4]: df.tail()
```

Out[4]:

	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

c. The number of unique users in the dataset.

```
In [5]: df.nunique()
```

```
Out[5]: user_id         290584
        timestamp       294478
        group                2
        landing_page         2
        converted            2
        dtype: int64
```

```
In [6]: df.user_id.nunique()
```

```
Out[6]: 290584
```

d. The proportion of users converted.

```
In [7]: df.converted.sum()/290584
```

```
Out[7]: 0.12126269856564711
```

e. The number of times when the "group" is `treatment` but "landing_page" is not a `new_page`.

```
In [8]: treat_old = df.query("group == 'treatment' and landing_page == 'old_page'")
        treat_new = df.query("group == 'control' and landing_page == 'new_page'")

        len(treat_old) + len(treat_new)
```

```
Out[8]: 3893
```

f. Do any of the rows have missing values?

```
In [9]: 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
```

1.0.2 ToDo 1.2

In a particular row, the **group** and **landing_page** columns should have either of the following acceptable values:

user_id	timestamp	group	landing_page	converted
XXXX	XXXX	`control`	`old_page`	X
XXXX	XXXX	`treatment`	`new_page`	X

It means, the `control` group users should match with `old_page`; and `treatment` group users should matched with the `new_page`. This is not true for all the users currently.

For the rows where `treatment` does not match with `new_page` or `control` does not match with `old_page`, we cannot be sure if such rows truly received the new or old wepage. Therefore, the next action will be to remove the inaccurate rows.

a. Create a new dataset doesn't contain these inaqurate rows. Store your new dataframe in **df2**.

```
In [10]: # Remove the inaccurate rows, and store the result in a new dataframe df2
         df2 = df.query("group == 'treatment' and landing_page == 'new_page'")
```

```
df2 = df2.append(df.query("group == 'control' and landing_page == 'old_page'"))
```

```
In [11]: # Double Check all of the incorrect rows were removed from df2 -
         # Output of the statement below should be 0
         df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sh
```

```
Out[11]: 0
```

1.0.3 ToDo 1.3

a. How many unique **user_id**s are in **df2**?

```
In [12]: df2.user_id.nunique()
```

```
Out[12]: 290584
```

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

```
In [13]: df2[df2['user_id'].duplicated()]
```

```
Out[13]:       user_id                   timestamp      group landing_page  converted
         2893   773192  2017-01-14 02:55:59.590927  treatment     new_page          0
```

c. Display the rows for the duplicate **user_id**?

```
In [14]: df2[df2['user_id'] == 773192]
```

```
Out[14]:       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**, from the **df2** dataframe.

```
In [15]: # Remove one of the rows with a duplicate user_id..

         df2 = df2.drop(2893)

         # Check again if the row with a duplicate user_id is deleted or not
         df2[df2['user_id'] == 773192]
```

```
Out[15]:       user_id                   timestamp      group landing_page  converted
         1899   773192  2017-01-09 05:37:58.781806  treatment     new_page          0
```

```
In [16]: df2.head()
```

```
Out[16]:    user_id                   timestamp      group landing_page  converted
         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
         6   679687  2017-01-19 03:26:46.940749  treatment     new_page          1
         8   817355  2017-01-04 17:58:08.979471  treatment     new_page          1
         9   839785  2017-01-15 18:11:06.610965  treatment     new_page          1
```

1.0.4 ToDo 1.4

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

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

> **Tip**: The probability you'll compute represents the overall "converted" success rate in the population and you may call it $p_{population}$.

```
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]: len(df2.query("group == 'control' and converted == '1'"))/len(df2.query("group == 'cont

         #basically, x in control and converted (divide by)
         # x in control
```

```
Out[18]: 0.1203863045004612
```

```
In [19]: #Or we could do it like this:

         control_prob = df2.query("group == 'control'")['converted'].mean()
         control_prob
```

```
Out[19]: 0.1203863045004612
```

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

```
In [20]: treatment_prob = df2.query("group == 'treatment'")['converted'].mean()
         treatment_prob
```

```
Out[20]: 0.11880806551510564
```

> **Tip**: The probabilities you've computed in the points (b). and (c). above can also be treated as conversion rate. Calculate the actual difference (`obs_diff`) between the conversion rates for the two groups. You will need that later.

```
In [21]: # Calculate the actual difference (obs_diff) between the conversion rates for the two g

         obs_diff = control_prob - treatment_prob
         obs_diff
```

```
Out[21]: 0.0015782389853555567
```

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

```
In [22]: df2.query('landing_page == "new_page"').shape[0]/df2.shape[0]
```

```
Out[22]: 0.5000619442226688

In [23]: #or we could do it like this
         len(df2.query('landing_page == "new_page"')) / len(df2)

Out[23]: 0.5000619442226688
```

e. Consider your results from parts (a) through (d) above, and explain below whether the new `treatment` group users lead to more conversions.

> **Based on the probabilities of `treatment_prob` and `control_prob`, I can say that there isn't much difference between whether the treatment group led to more conversions than the control group. The control group has slightly more conversions, but this difference seems to be negligible**

Part II - A/B Test

Since a timestamp is associated with each event, you could run a hypothesis test continuously as long as you observe the events.

However, then the hard questions would be: - 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.

1.0.5 ToDo 2.1

> Recall that you just calculated that the "converted" probability (or rate) for the old page is *slightly* higher than that of the new page (ToDo 1.4.c).

What should be your null and alternative hypotheses (

$H_0 = p_{old} = p_{new}$

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

1.0.6 ToDo 2.2 - Null Hypothesis H_0 Testing

Under the null hypothesis H_0, assume that p_{new} and p_{old} are equal. Furthermore, assume that p_{new} and p_{old} both are equal to the **converted** success rate in the `df2` data regardless of the page. So, our assumption is:

$p_{new} = p_{old} = p_{population}$

In this section, I will:

- Simulate (bootstrap) sample data set for both groups, and compute the "converted" probability p for those samples.
- Use a sample size for each group equal to the ones in the `df2` data.
- Compute the difference in the "converted" probability for the two samples above.

- Perform the sampling distribution for the "difference in the converted probability" between the two simulated-samples over 10,000 iterations; and calculate an estimate.

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

```
In [24]: p_new = df2.converted.mean()
         p_new

Out[24]: 0.11959708724499628
```

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

```
In [25]: # remember, the assumption is that p_new = p_old
         p_old = df2.converted.mean()
```

c. What is n_{new}, the number of individuals in the treatment group? *Hint*: The treatment group users are shown the new page.

```
In [26]: n_new = df2.query('landing_page == "new_page"').shape[0]
         n_new

Out[26]: 145310
```

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

```
In [27]: n_old = df2.query('group == "control"').shape[0]
         n_old

Out[27]: 145274
```

e. Simulate Sample for the `treatment` Group Simulate n_{new} transactions with a conversion rate of p_{new} under the null hypothesis.

```
In [28]: # Simulate a Sample for the treatment Group
         new_page_converted = np.random.choice([0, 1], n_new, p = [p_new, 1-p_new])
         new_page_converted

Out[28]: array([1, 1, 0, ..., 0, 1, 1])
```

f. Simulate Sample for the `control` Group Simulate n_{old} transactions with a conversion rate of p_{old} under the null hypothesis. Store these n_{old} 1's and 0's in the `old_page_converted` numpy array.

```
In [29]: # Simulate a Sample for the control Group
         old_page_converted = np.random.choice([0, 1], n_old, p = [p_old, 1-p_old])
```

g. Find the difference in the "converted" probability $(p'_{new} - p'_{old})$ for your simulated samples from the parts (e) and (f) above.

```
In [30]: print(new_page_converted.mean() - old_page_converted.mean())
```

```
0.000277414552774
```

h. Sampling distribution Re-create `new_page_converted` and `old_page_converted` and find the $(p'_{new} - p'_{old})$ value 10,000 times using the same simulation process you used in parts (a) through (g) above.

Store all $(p'_{new} - p'_{old})$ values in a NumPy array called `p_diffs`.

```
In [31]: p_diffs = []
         for i in range(10000):

             # 1st parameter dictates the choices you want.  In this case [1, 0]
             p_new_10k = np.random.choice([1, 0],n_new,replace = True,p = [p_new, 1-p_new])
             p_old_10k = np.random.choice([1, 0],n_old,replace = True,p = [p_old, 1-p_old])
             p_new_10k = p_new_10k.mean()
             p_old_10k = p_old_10k.mean()
             p_diffs.append(p_new_10k-p_old_10k)
```

```
In [32]: p_diffs = np.array(p_diffs)
```

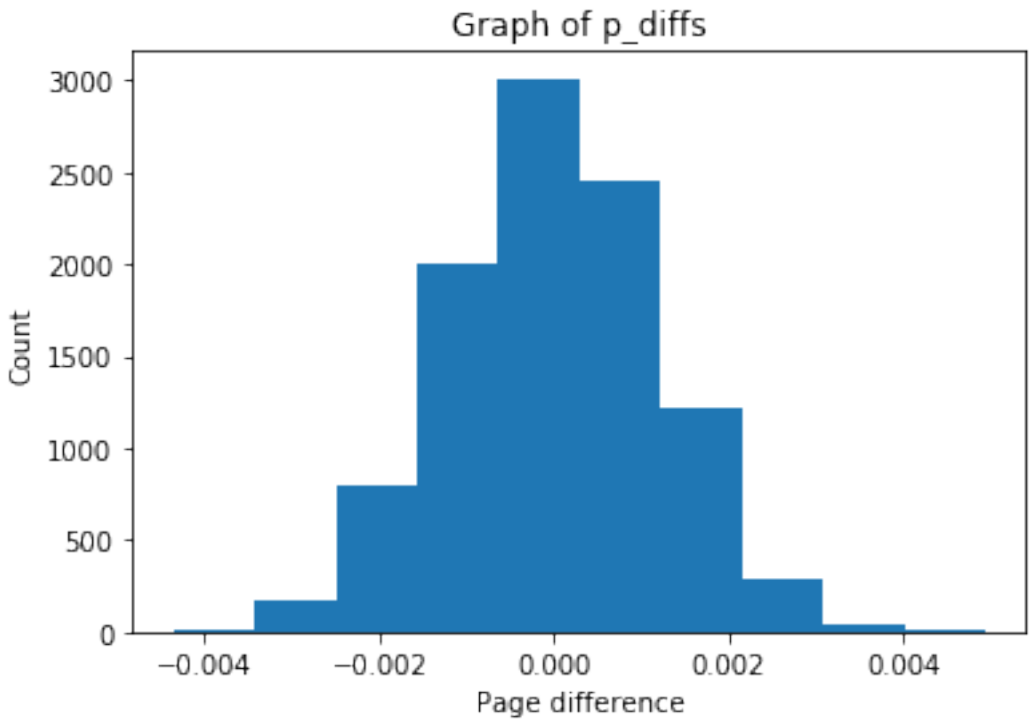
```
In [33]: len(p_diffs)
```

```
Out[33]: 10000
```

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

```
In [34]: #plt.hist(p_diffs);
         plt.hist(p_diffs)
         plt.title('Graph of p_diffs')#title of graphs
         plt.xlabel('Page difference') # x-label of graphs
         plt.ylabel('Count') # y-label of graphs
```

```
Out[34]: Text(0,0.5,'Count')
```



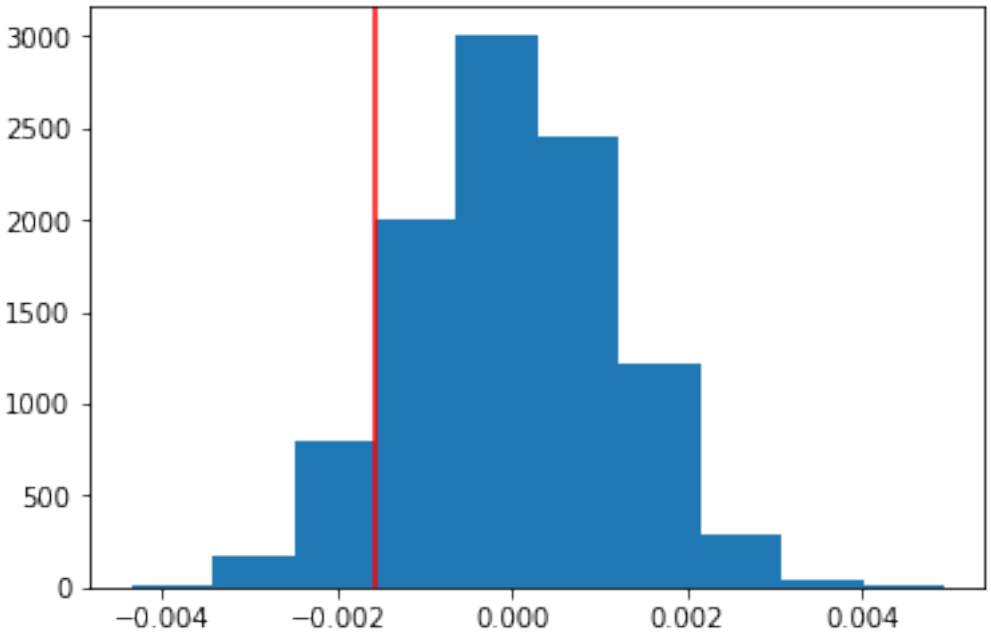
In [35]:
```
obs_diff = treatment_prob - control_prob

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

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

p_val = low_prob + high_prob
print(p_val)
```

```
0.19
```



j. What proportion of the **p_diffs** are greater than the actual difference observed in the `df2` data?

```
In [36]: (p_diffs > obs_diff).mean()
```

```
Out[36]: 0.90339999999999998
```

k. Please explain in words what you have just computed in part **j** above.
- What is this value called in scientific studies?
- What does this value signify in terms of whether or not there is a difference between the new and old pages?

> **The value calculated is the p-value. The p-value here is about `0.9` and this is higher than the type error rate of `0.05`. This p-value tells us that about `90%` of the population is highter than the actual difference observed. When a p-value is less than `0.05`, we are to reject the null hypothesis, because that is a statistically significant result. In this case, the p-value is higher than `0.05`, and as such, we cannot reject the null hypothesis that says that `p_new = p_old`. This means that we cannot say that the new page is better than the old page.**

l. Using Built-in Methods for Hypothesis Testing 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 walk-through of the ideas that are critical to correctly thinking about statistical significance.

Fill in the statements below to calculate the: - `convert_old`: number of conversions with the old_page - `convert_new`: number of conversions with the new_page - `n_old`: number of individuals who were shown the old_page - `n_new`: number of individuals who were shown the new_page

```
In [37]: # number of conversions with the old_page
         convert_old = df2.query('converted == 1 and landing_page == "old_page"').shape[0]

         # number of conversions with the new_page
         convert_new = df2.query('converted == 1 and landing_page == "new_page"').shape[0]

         # number of individuals who were shown the old_page
         n_old = len(df2.query('landing_page == "old_page"'))

         # number of individuals who received new_page
         n_new = len(df2.query('landing_page == "new_page"'))
```

1.0.7 About the two-sample z-test

Next step is to make a decision to reject or fail to reject the null hypothesis based on comparing these two values: - Z_{score} - Z_{α} or $Z_{0.05}$, also known as critical value at 95% confidence interval. $Z_{0.05}$ is 1.645 for one-tailed tests, and 1.960 for two-tailed test. You can determine the Z_{α} from the z-table manually.

Decide if your hypothesis is either a two-tailed, left-tailed, or right-tailed test. Accordingly, reject OR fail to reject the null based on the comparison between Z_{score} and Z_{α}.

```
In [38]: import statsmodels.api as sm
         # ToDo: Complete the sm.stats.proportions_ztest() method arguments
         z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new
         print(z_score, p_value)
```

```
/opt/conda/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: FutureWarning: The panda
  from pandas.core import datetools


1.31092419842 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.**?

- The z-score here is about `1.31` and this is a positive value.
- In a right-tailed test, we reject null if $Z_{score} > Z_{\alpha}$ or less than -(Z_{α}). The Z_{α} (the critical value at 95% confidence interval) is `1.645` for a one-tailed test.
- In this test, $Z_{score} < Z_{\alpha}$ (`1.31 < 1.645`) and as such, we fail to reject the null hypothesis.
- With a positive z-score, we can also deduce that the z-score 1.31 standard deviations higher than the mean.
- The p-value obtained here is also similar to the p-value obtained before, i.e., ~`0.9`, and as such, we fail to reject the null hypothesis that says that `p_new = p_old`. This means that we cannot say that the new page is better than the old page.

Part III - A regression approach

1.0.8 ToDo 3.1

In this final part, I will show that the result achieved in the A/B test in Part II above can also be achieved by performing regression.

a. Since each row in the `df2` data 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** library to fit the regression model you specified in part **a.** above to see if there is a significant difference in conversion based on the page-type a customer receives. However, you first need to create the following two columns in the `df2` dataframe: 1. `intercept` - It should be `1` in the entire column. 2. `ab_page` - It's a dummy variable column, having a value `1` when an individual receives the **treatment**, otherwise `0`.

```
In [39]: df2.head()
```

Out[39]:

	user_id	timestamp	group	landing_page	converted
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
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1

```
In [40]: df_logreg = df2.copy()
         df_logreg.head()
```

Out[40]:

	user_id	timestamp	group	landing_page	converted
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
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1

```
In [41]: # addding an intercept
         df_logreg['intercept'] = 1

         # creating dummy columns
         df_logreg['ab_page'] = pd.get_dummies(df_logreg['group'])['treatment']


         # checking again
         df_logreg.head()
```

Out[41]:

	user_id	timestamp	group	landing_page	converted	\
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	
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1	
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1	

```
9   839785  2017-01-15 18:11:06.610965  treatment     new_page          1

   intercept  ab_page
2          1        1
3          1        1
6          1        1
8          1        1
9          1        1
```

c. Use **statsmodels** to instantiate your regression model on the two columns you created in part (b). above, then fit the model to predict whether or not an individual converts.

```
In [42]: log_mod =sm.Logit(df_logreg['converted'], df_logreg[['intercept', 'ab_page']])
         results=log_mod.fit()

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

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

```
In [43]: results.summary2()

Out[43]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                Results: Logit
         ================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2022-05-31 11:22 AIC:              212780.3502
         No. Observations:   290584           BIC:              212801.5095
         Df Model:           1                Log-Likelihood:   -1.0639e+05
         Df Residuals:       290582           LL-Null:          -1.0639e+05
         Converged:          1.0000           Scale:            1.0000
         ----------------------------------------------------------------
                       Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
         ----------------------------------------------------------------
         intercept    -1.9888    0.0081  -246.6690  0.0000  -2.0046  -1.9730
         ab_page      -0.0150    0.0114    -1.3109  0.1899  -0.0374   0.0074
         ================================================================

         """
```

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

The p-value associated with ab_page is 0.1899. The null and alternative hypothesis associated with the regression model are as follows:

$H_0 = p_{old} - p_{new} = 0$ $H_1 = p_{new} - p_{old}$ 0

In part II, the test was a one sided (right tailed test), while the logistic regression approach is a two tailed test. Inspite of their differences, the summary table above shows the same result for the `ab_page` z-test, i.e., `1.3109` Another thing to note is that the p-value for ab_page is 0.1899, and this is higher than the 0.05, meaning that we can fail to reject the null hypothesis in this case. This means that we cannot say that the new page is better than the old page.

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?

```
In [44]: # duration of the experiment
         duration = np.array(pd.to_datetime(df.timestamp).sort_values(ascending=True))
         td = duration[-1] - duration[0]
         days = td.astype('timedelta64[D]')
         days / np.timedelta64(1, 'D')
```

```
Out[44]: 21.0
```

- The duration of the project may influence the results seen so far. 21 days (as evaluated above) can be considered a short time to run the A/B tests, as it may allow change aversion effects to occur.
- The countries where the page is opened may also affect individual converts where language barrier is a challenge.

g. Adding countries Now along with testing if the conversion rate changes for different pages, I'll also add an effect based on which country a user lives in.

1. I will read in the **countries.csv** dataset and merge together your `df2` datasets on the appropriate rows.
2. Does it appear that country had an impact on conversion? To answer this question, I'll consider the three unique values, `['UK', 'US', 'CA']`, in the `country` column. Create dummy variables for these country columns.

```
In [45]: # Read the countries.csv
         df_countries = pd.read_csv('countries.csv')
         df_countries.head()
```

```
Out[45]:    user_id country
         0   834778      UK
         1   928468      US
         2   822059      UK
         3   711597      UK
         4   710616      UK
```

```
In [46]: # Join with the df2 dataframe
         df_merged = df2.join(df_countries.set_index('user_id'), on='user_id')
         # checking...
         df_merged.head()
```

```
Out[46]:    user_id                   timestamp      group landing_page  converted  \
          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
          6   679687  2017-01-19 03:26:46.940749  treatment     new_page          1
          8   817355  2017-01-04 17:58:08.979471  treatment     new_page          1
          9   839785  2017-01-15 18:11:06.610965  treatment     new_page          1

            country
          2      US
          3      US
          6      CA
          8      UK
          9      CA
```

```
In [47]: # Create the necessary dummy variables
         df_merged[['canada','uk','us']] = pd.get_dummies(df_merged['country'])

         df_merged.head()
```

```
Out[47]:    user_id                   timestamp      group landing_page  converted  \
          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
          6   679687  2017-01-19 03:26:46.940749  treatment     new_page          1
          8   817355  2017-01-04 17:58:08.979471  treatment     new_page          1
          9   839785  2017-01-15 18:11:06.610965  treatment     new_page          1

            country  canada  uk  us
          2      US       0   0   1
          3      US       0   0   1
          6      CA       1   0   0
          8      UK       0   1   0
          9      CA       1   0   0
```

```
In [48]: # Using Canada as baseline, we'll drop Canada..
         df_merged.drop(['canada'], axis=1, inplace=True)
```

h. Fit your model and obtain the results

Tip: Conclusions should include both statistical reasoning, and practical reasoning for the situation.

Hints: - Look at all of p-values in the summary, and compare against the Type I error rate (0.05). - Can you reject/fail to reject the null hypotheses (regression model)? - Comment on the effect of page and country to predict the conversion.

```
In [49]: # Fit your model

         # creating intercept
         df_merged['intercept'] = 1
```

```
        logit_mod = sm.Logit(df_merged['converted'], df_merged[['intercept','us','uk']])
        results = logit_mod.fit()

Optimization terminated successfully.
        Current function value: 0.366116
        Iterations 6


In [50]: # and summarize the results

        results.summary2()

Out[50]: <class 'statsmodels.iolib.summary2.Summary'>
        """
                                Results: Logit
        =================================================================
        Model:              Logit            No. Iterations:   6.0000
        Dependent Variable: converted        Pseudo R-squared: 0.000
        Date:               2022-05-31 11:22 AIC:              212780.8333
        No. Observations:   290584           BIC:              212812.5723
        Df Model:           2                Log-Likelihood:   -1.0639e+05
        Df Residuals:       290581           LL-Null:          -1.0639e+05
        Converged:          1.0000           Scale:            1.0000
        -----------------------------------------------------------------
                       Coef.   Std.Err.     z      P>|z|    [0.025   0.975]
        -----------------------------------------------------------------
        intercept     -2.0375    0.0260  -78.3639  0.0000  -2.0885  -1.9866
        us             0.0408    0.0269    1.5178  0.1291  -0.0119   0.0935
        uk             0.0507    0.0284    1.7863  0.0740  -0.0049   0.1064
        =================================================================

        """

In [51]: # Calculating the exponential of the results for better interpretation

        1/np.exp(0.0408), np.exp(0.0507)

Out[51]: (0.96002111497165088, 1.0520072437650141)
```

Results Summary. Based on the results above, the people in the US are 0.96 times more likely to convert compared to users in Canada. Also, the people in the UK are 0.507 times more likely to convert compared to users in Canada. This result however may nor be statistically significant as both the US and the UK have produced p-values higher than 0.05. This is similar to the results from the initial p-value, and z-score where we filed to reject the null hypothesis

2 Conclusion

The aim of this project was to understand and determine whether a company should implement a new page or keep the old one. To achieve this we used: - Probability test. - A/B test. - Regression.

2.0.1 Probability Test

- With the probability test, we found out that the probability that an individual received the new page was `0.500`. Meaning that the population had approximately a 50% chance in receiving the new page
- We also found that there was not mucg difference between the probabilty that an individual in either the treatment group or control group converted.

2.0.2 A/B Test

- In the A/B Tests, the hypothesis was set up to determine if the new page gives better conversion or not.
- A/B test results showed a p-value of approximately `0.9` which is higher than the type error rate of `0.05`.
- With this result, we failed to reject the null hypothesis; $H_0 = p_{old} = p_{new}$.
- The in-built z-test (right tailed) was also computed. With a z-score of `1.31`, we failed to reject the null hypothesis, just as we did previously with the p-value results.

2.0.3 Regression Test

- A logistic regression test was conducted to explore two possible outcomes, i.e, to determine if the new page is better or not.
- The logistic regression results produced the same z-score as the one obtained in the A/B test. The logistic regression also produced a p-value of `0.190`, higher than the type error value of `0.05`, and as such, we again failed to reject the null hypothesis.
- Countries were considered to have probable impact on conversion rates and as such were added to the model in another logistic regression testing.
- The results were statistically insignificant and as such, it cannot be said that the countries have any impact on the conversion rate

2.0.4 Considerations

- The duration of the test was also considered as a probable factor that may impact the results of the test.
- The duration of the test was determined to be 21 days. 21 days can be considered a relatively short time to run the A/B test.
- This short time may increase the possibility for change aversion effect to occur, where users may give unfair advantate to the old page.

2.1 Resources:

- Statistics, courtesy www.stats.libretexts.org
- Joining Tables

- Investopia: One Sample Z-Test Example

Final Check!

Congratulations! You have reached the end of the A/B Test Results project! You should be very proud of all you have accomplished!

> **Tip**: Once you are satisfied with your work here, check over your notebook to make sure that it satisfies all the specifications mentioned in the rubric. You should also probably remove all of the "Hints" and "Tips" like this one so that the presentation is as polished as possible.

Submission You may either submit your notebook through the "SUBMIT PROJECT" button at the bottom of this workspace, or you may work from your local machine and submit on the last page of this project lesson.

1. Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).
2. Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.
3. Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

```
In [52]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[52]: 0
```

```
In [ ]:
```