Analyze_ab_test_results_notebook

May 7, 2020

0.1 Analyze A/B Test Results

You may either submit your notebook through the workspace here, or you may work from your local machine and submit through the next page. Either way assure that your code passes the project RUBRIC. **Please save regularly.**

This project will assure you have mastered the subjects covered in the statistics lessons. The hope is to have this project be as comprehensive of these topics as possible. Good luck!

0.2 Table of Contents

Introduction

A/B tests are very commonly performed by data analysts and data scientists. It is important that you get some practice working with the difficulties of these

For this project, you will be working to understand the results of an A/B test run by an e-commerce website. Your goal is to work through this notebook to help the company understand if they should implement the new page, keep the old page, or perhaps run the experiment longer to make their decision.

As you work through this notebook, follow along in the classroom and answer the corresponding quiz questions associated with each question. The labels for each classroom concept are provided for each question. This will assure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the criteria. As a final check, assure you meet all the criteria on the RUBRIC.

Part I - Probability

To get started, let's import our libraries.

```
In [191]: 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 [192]: #Read dataset
          df = pd.read_csv("ab_data.csv")
          df.head()
```

Out[192]:

	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 [193]: df.shape[0]
```

```
Out[193]: 294478
```

There are 294478 rows in the dataset.

c. The number of unique users in the dataset.

```
In [194]: #unique elements
          df['user_id'].nunique()
```

```
Out[194]: 290584
```

There are 290584 unique users

d. The proportion of users converted.

```
In [196]: #mean conversions
          df.converted.mean()
```

```
Out[196]: 0.11965919355605512
```

The proportion of user coverted is 11.9%

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

```
In [197]: no_match1 = df.query('group == "control" and landing_page == "new_page"')
          len(no_match1)
```

```
Out[197]: 1928
```

```
In [137]: no_match2 = df.query('group == "treatment" and landing_page == "old_page"')
          len(no_match2)
```

```
Out[137]: 1965

In [138]: # Total
          match = len(no_match1) + len(no_match2)
          match

Out[138]: 3893
```

3893 times the new_page and treatment don't match.

f. Do any of the rows have missing values?

```
In [198]: #find null values
          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
```

There is no row with a missing value.

2. For the rows where **treatment** does not match with **new_page** or **control** does not match with **old_page**, we cannot be sure if this row truly received the new or old page. Use **Quiz 2** in the classroom to figure out 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 [199]: #copy df to df2
          df2=df.copy()
          df2.drop(df.query("(group == 'treatment' and landing_page == 'old_page') or (group ==

In [201]: # Double Check all of the correct rows were removed - this should be 0
          df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].s

Out[201]: 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 [202]: uni = df2.user_id.nunique()
          print("Unique user_ids: {}".format(uni))
```

```
Unique user_ids: 290584
```

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

```
In [203]: #Repeated user ids
          df2[df2["user_id"].duplicated()]['user_id']

Out[203]: 2893    773192
          Name: user_id, dtype: int64
```

The repeated user id in df2 is 773192

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

```
In [204]: 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

```
In [205]: df2[df2.duplicated(['user_id'], keep=False)]
```

	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 [206]: df2=df2.drop_duplicates(['user_id'])

In [207]: df2[df2['user_id'] == 773192]
```

	user_id	timestamp	group	landing_page	converted
1899	773192	2017-01-09 05:37:58.781806	treatment	new_page	0

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?

```
In [210]: #probability of individual converted
          df2['converted'].mean()

Out[210]: 0.11959708724499628
```

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

```
In [211]: #probability of individuals in control group which converted
          df2[df2['group']=='control']['converted'].mean()

Out[211]: 0.1203863045004612
```

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

```
In [217]: #probability of individuals in treatment group which converted
          df2[df2['group'] == 'treatment']['converted'].mean()

Out[217]: 0.11880806551510564
```

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

```
In [218]: #probability that individual received new page
          len(df2[df2['landing_page'] == 'new_page'])/len(df2)

Out[218]: 0.5000619442226688
```

e. Consider your results from parts (a) through (d) above, and explain below whether you think there is sufficient evidence to conclude that the new treatment page leads to more conversions.

Your answer goes here.

- The probability of an individual getting converted in the treatment group is 0.118 and on the control group is 0.120. Though the probability of an individual getting converted has increased but by a very little change of 0.002. We can say that the old page does better but by a very small margin. 50% of the individuals received the new page. Also, as we did not take all the other elements which could have led to this change, we cannot conclude which page converts more individuals based on the numbers above.

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.

Put your answer here.

Hypothesis: The null hypothesis will is set assuming the old page is better than the new page and it would be rejected if the conversion rate of the new page is heigher than the conversion rate of the old page:

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

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

Where p_{new} & p_{old} are the converted rates for the old and new pages.

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 **conversion rate** for p_{new} under the null?

```
In [152]: P_new = df2['converted'].mean()
          P_new

Out[152]: 0.11959708724499628
```

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

```
In [219]: P_old = df2['converted'].mean()
          P_old

Out[219]: 0.11959708724499628
```

c. What is n_{new}, the number of individuals in the treatment group?

```
In [220]: n_new = len(df2.query('landing_page == "new_page"'))
          n_new

Out[220]: 145310
```

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

```
In [224]: n_old = len(df2.query('landing_page == "old_page"'))
          n_old

Out[224]: 145274
```

e. Simulate n_{new} transactions with a conversion rate of p_{new} under the null. Store these n_{new} 1's and 0's in **new_page_converted**.

```
In [225]: new_page_converted = np.random.binomial(n_new,P_new)
          new_page_converted

Out[225]: 17430
```

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

```
In [228]: old_page_converted = np.random.binomial(n_old,P_old)
          old_page_converted
```

```
Out[228]: 17481
```

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

```
In [229]: diff = (new_page_converted/n_new) - (old_page_converted/n_old)
          diff
```

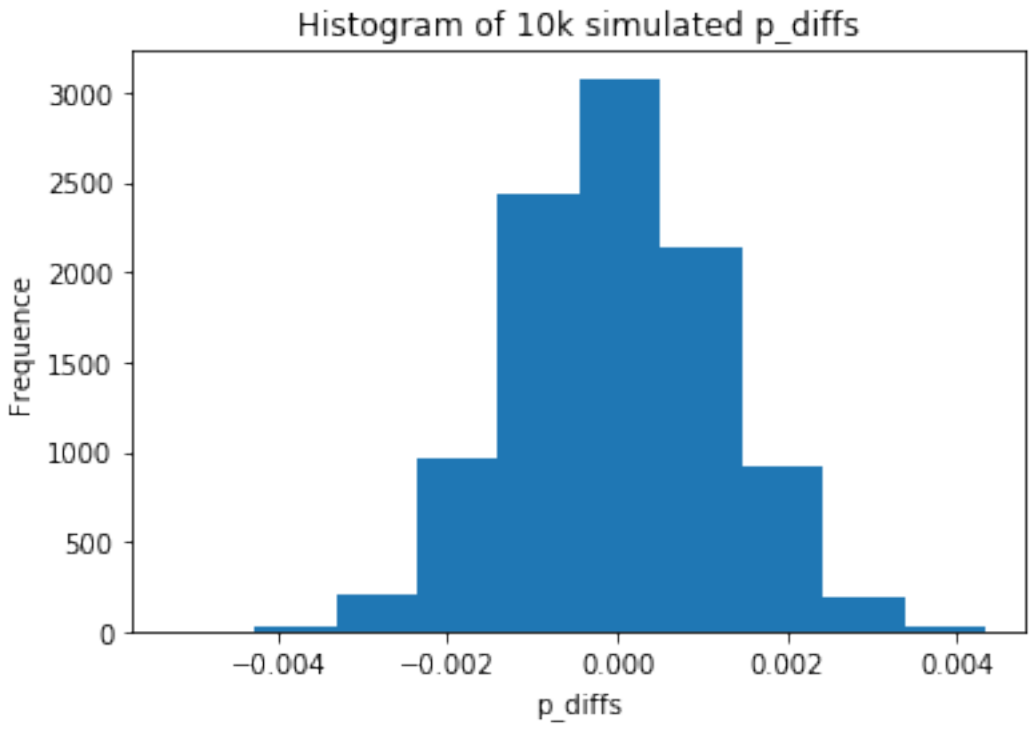
```
Out[229]: -0.00038078538642410953
```

h. Create 10,000 p_{new} - p_{old} values using the same simulation process you used in parts (a) through (g) above. Store all 10,000 values in a NumPy array called **p_diffs**.

```
In [230]: #Create 10,000  -  values
          p_diffs = []
          for _ in range(10000):
              new_page = np.random.binomial(n_new,P_new)
              old_page = np.random.binomial(n_old,P_old)
              diffs = (new_page/n_new) - (old_page/n_old)
              p_diffs.append(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.

```
In [231]: plt.hist(p_diffs)
          plt.xlabel('p_diffs')
          plt.ylabel('Frequence')
          plt.title("Histogram of 10k simulated p_diffs");
```



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

```
In [232]: #Calculate the difference in mean in the main data
          df_mean_old = df.query('group =="control"').converted.mean()
          df_mean_new = df.query('group =="treatment"').converted.mean()
          df_diff = df_mean_new - df_mean_old

In [233]: #convert into array
          df_diffs = np.array(df_diff)

          #Calculating the mean of the difference
          (p_diffs>df_diffs).mean()

Out[233]: 0.89190000000000003
```

k. Please explain using the vocabulary you've learned in this course 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?

Put your answer here. * Here we have calculate the p value. As the p value is quite big we failed to reject the null hypothesis. * We can conclude that the new page does not lead to more conversions.

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 [235]: import statsmodels.api as sm
          from scipy import stats
          stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)

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

          convert_old, convert_new, n_old, n_new

Out[235]: (17489, 17264, 145274, 145310)
```

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

```
In [236]: z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_ne
          z_score, p_value

Out[236]: (1.3109241984234394, 0.90505831275902449)
```

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.**?

```
In [237]: from scipy.stats import norm

          z_score = norm.cdf(z_score)
          print(z_score)
          print(norm.ppf(1-(0.05)/2))

0.905058312759
1.95996398454
```

Put your answer here.

Since our z score 1.31 lies between the critical range of +-1.96. we fail to reject the null hypothesis. p-value determines the significance of the results. The values are different from parts j. and k. but it still suggests that there is no statistically significant difference betweem the new and the old page.

Part III - A regression approach

1. In this final part, you will see that the result you achieved in the A/B test in Part II above 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?

Put your answer here.
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 in df2 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 [238]: df2['intercept'] = 1
          df2[['ab_page', 'old_page']] = pd.get_dummies(df2['landing_page'])
          df2.head()

Out[238]:    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  old_page
0          1        0         1
1          1        0         1
2          1        1         0
3          1        1         0
4          1        0         1
```

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

```
In [239]: logit_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
          results = logit_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 [240]: results.summary()

Out[240]: <class 'statsmodels.iolib.summary.Summary'>
          """
                                Logit Regression Results
          ==============================================================================
          Dep. Variable:              converted   No. Observations:               290584
          Model:                          Logit   Df Residuals:                   290582
          Method:                           MLE   Df Model:                            1
          Date:                Thu, 07 May 2020   Pseudo R-squ.:               8.077e-06
          Time:                        15:20:19   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**?

Answer

The p-value calculated here is 0.190. Logit regression is based on two tailed test and as 0.190 is still greater than 0.05 we are unable to reject the null hypothesis.

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?

Answer

The conversion rate might be effected by other demographic or geographical factors. We could find out the details of the users that are getting converted and the buying patterns in order to make changes to the old page and make it more effective.

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

```
In [182]: countries_df = pd.read_csv('./countries.csv')

In [183]: #joining dataframes
          df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
          df_new.head()
```

Out[183]:

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

```
In [184]: df_new.country.unique()

Out[184]: array(['UK', 'US', 'CA'], dtype=object)

In [185]: # Create the dummy variables
          df_new[['CA', 'UK', 'US']] = pd.get_dummies(df_new['country'])
          df_new.head()
```

Out[185]:
```
         country                   timestamp      group landing_page  \
user_id
834778       UK  2017-01-14 23:08:43.304998    control     old_page
928468       US  2017-01-23 14:44:16.387854  treatment     new_page
822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
711597       UK  2017-01-22 03:14:24.763511    control     old_page
710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

         converted  intercept  ab_page  old_page  CA  UK  US
user_id
834778           0          1        0         1   0   1   0
928468           0          1        1         0   0   0   1
822059           1          1        1         0   0   1   0
711597           0          1        0         1   0   1   0
710616           0          1        1         0   0   1   0
```

In [186]:
```
# Train the model
log_mod = sm.Logit(df_new['converted'], df_new[['intercept', 'UK', 'US']])
result = log_mod.fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366116
         Iterations 6
```

Out[186]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290581
Method:                           MLE   Df Model:                            2
Date:                Thu, 07 May 2020   Pseudo R-squ.:               1.521e-05
Time:                        14:49:10   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.1984
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -2.0375      0.026    -78.364      0.000      -2.088      -1.987
UK             0.0507      0.028      1.786      0.074      -0.005       0.106
US             0.0408      0.027      1.518      0.129      -0.012       0.093
==============================================================================
"""
```

Conclusion

As in this case also p-value is greater than 0.05. We failed to reject the null hypothesis and we can conclude that countries makes a very little difference on the conversion rate.

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 [241]: #New columns showing interactions between countries and page
          df_new['US_page'] = df_new['US'] * df_new['ab_page']
          df_new['UK_page'] = df_new['UK'] * df_new['ab_page']

In [243]: #summary results
          log_res = sm.Logit(df_new['converted'], df_new[['intercept', 'ab_page', 'US', 'UK', 'U
          results = log_res.fit()
          results.summary()

Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 6


Out[243]: <class 'statsmodels.iolib.summary.Summary'>
          """
                                    Logit Regression Results
          ==============================================================================
          Dep. Variable:              converted   No. Observations:               290584
          Model:                          Logit   Df Residuals:                   290578
          Method:                           MLE   Df Model:                            5
          Date:                Thu, 07 May 2020   Pseudo R-squ.:              3.482e-05
          Time:                        15:24:44   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
          UK             0.0118      0.040      0.296      0.767      -0.066       0.090
          US_page        0.0469      0.054      0.872      0.383      -0.059       0.152
          UK_page        0.0783      0.057      1.378      0.168      -0.033       0.190
          ==============================================================================
          """
```

Here again, looking at the p-values we can say that the interaction between page and country have no significant effects on conversion.

Conclusions:

After considering the whole test, we have no strong evidence to conclude that new page results in more conversions as compared to the old one.

Although there might be other factors which could lead to more conversions but with the indormation and data in hand we fail to reject the null hypothesis.We found that the performance of the old page was better to some extent and we should drop the idea of introducing a new page.
Directions to Submit

> 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).
>
> 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.
>
> 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 [190]: from subprocess import call
          call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[190]: 0
```

```
In [ ]:
```