Analyze_ab_test_results_notebook

June 26, 2020

0.1 Analyze A/B Test Results

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

0.2 Table of Contents

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 [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]: df.shape
```

```
Out[3]: (294478, 5)
```

c. The number of unique users in the dataset.

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

```
Out[4]: user_id         290584
        timestamp       294478
        group                2
        landing_page         2
        converted            2
        dtype: int64
```

d. The proportion of users converted.

```
In [5]: (df.converted == 0).mean(), (df.converted == 1).mean(),
```

```
Out[5]: (0.88034080644394486, 0.11965919355605512)
```

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

```
In [6]: df.groupby(['group', 'landing_page'], as_index=False).count()
```

Out[6]:

	group	landing_page	user_id	timestamp	converted
0	control	new_page	1928	1928	1928
1	control	old_page	145274	145274	145274
2	treatment	new_page	145311	145311	145311
3	treatment	old_page	1965	1965	1965

```
In [7]: df[(df['group'] == 'treatment') != (df['landing_page'] == 'new_page')].shape
```

```
Out[7]: (3893, 5)
```

f. Do any of the rows have missing values?

```
In [8]: df.isnull().sum()
```

```
Out[8]: user_id         0
        timestamp       0
        group           0
        landing_page    0
        converted       0
        dtype: int64
```

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 [9]: treat = df.query('group == "treatment"').query('landing_page == "old_page"')
        treat.head()
```

```
Out[9]:      user_id                   timestamp      group landing_page  converted
        308   857184  2017-01-20 07:34:59.832626  treatment     old_page          0
        327   686623  2017-01-09 14:26:40.734775  treatment     old_page          0
        357   856078  2017-01-12 12:29:30.354835  treatment     old_page          0
        685   666385  2017-01-23 08:11:54.823806  treatment     old_page          0
        713   748761  2017-01-10 15:47:44.445196  treatment     old_page          0
```

```
In [10]: control = df.query('group == "control"').query('landing_page == "new_page"')
         control.head()
```

```
Out[10]:      user_id                   timestamp    group landing_page  converted
         22    767017  2017-01-12 22:58:14.991443  control     new_page          0
         240   733976  2017-01-11 15:11:16.407599  control     new_page          0
         490   808613  2017-01-10 21:44:01.292755  control     new_page          0
         846   637639  2017-01-11 23:09:52.682329  control     new_page          1
         850   793580  2017-01-08 03:25:33.723712  control     new_page          1
```

```
In [11]: invers = control.append(treat)
         df2 = df.drop(invers.index)
         df2.shape
```

```
Out[11]: (290585, 5)
```

```
In [12]: # Double Check all of the correct rows were removed - this should be 0
         df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sh
```

```
Out[12]: 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 [13]: df2['user_id'].nunique()
```

```
Out[13]: 290584
```

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

```
In [14]: duplicat_row = df2['user_id'].duplicated()
         duplicat_row.sum()
```

```
Out[14]: 1
```

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

```
In [15]: df2[duplicat_row]
```

```
Out[15]:       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**.

```
In [16]: df2.drop(df2[duplicat_row].index, inplace=True)
```

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

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

```
In [17]: (df2.converted == 1).mean()
```

```
Out[17]: 0.11959708724499628
```

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

```
In [18]: (df2.query('group == "control"').converted == 1).mean()
```

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

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

```
In [19]: (df2.query('group == "treatment"').converted == 1 ).mean()
```

```
Out[19]: 0.11880806551510564
```

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

```
In [20]: (df2.landing_page == 'new_page').mean()
```

```
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.

Your answer goes here. - note treatment group is equlse to new page and control group is equlse to old page

- The propability of individual landing on new page is coverd by 50%
- The total propability of individual converting is coverd by 11.96%
- The propabilty of individual converting in `old page` is `12.03%` hight than the probality of individuals converting in the `new page` which is `11.88%`

conclusions - This brings use to a simpson pardox which we assumed that all individual landing on the new page are 50% equal to individual landing on the old page

- This now brings a reality that the conversion rate of the old page is not equal but gretter than the new page

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.

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

$$H_1 : p_{new} - p_{old} > 0$$

`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?

```
In [21]: sample_size = df2.query('group == "treatment"').shape[0]

         new_page_sample = df2.sample(sample_size)

In [22]: p_new = (new_page_sample.converted == 1).mean()
         p_new
```

```
Out[22]: 0.11982657766155116
```

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

```
In [23]: sample_size = df2.query('group == "control"').shape[0]

         old_page_sample = df2.sample(sample_size)
```

```
In [24]: p_old = (old_page_sample.converted == 1).mean()
         p_old
```

```
Out[24]: 0.12020733235128103
```

c. What is n_{new}?

```
In [25]: new_page_sample.shape[0]
```

```
Out[25]: 145310
```

d. What is n_{old}?

```
In [26]: old_page_sample.shape[0]
```

```
Out[26]: 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 [27]: new_page_converted = new_page_sample['converted'].sample(new_page_sample.shape[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**.

```
In [28]: old_page_converted = old_page_sample['converted'].sample(old_page_sample.shape[0])
```

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

```
In [29]: obs_diffs = df2.query('group == "treatment"').converted.mean() - df2.query('group == "c
```

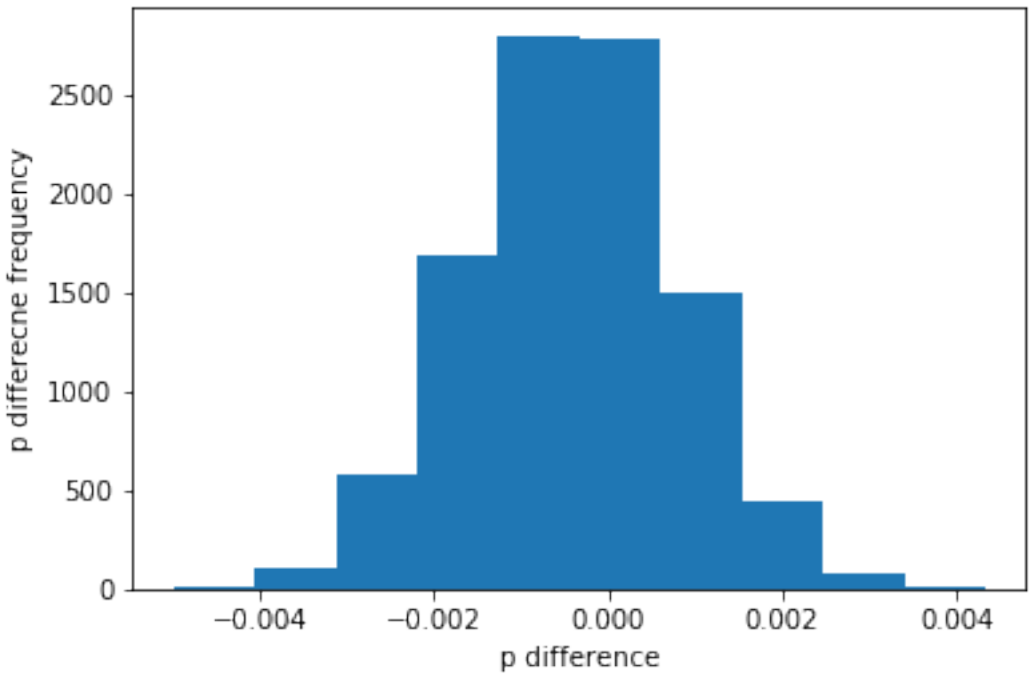
h. Simulate 10,000 p_{new} - p_{old} values using this same process similarly to the one you calculated in parts **a. through g.** above. Store all 10,000 values in a numpy array called **p_diffs**.

```
In [30]: p_diffs = []

         new_page_converted = np.random.binomial(new_page_sample.shape[0], p_new, 10000)/new_pag
         old_page_converted = np.random.binomial(old_page_sample.shape[0], p_old, 10000)/old_pag
         p_diffs.append(new_page_converted-old_page_converted)
```

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 [31]: p_diffs = np.array(p_diffs)
         p_diffs = p_diffs.reshape(10000, 1)
         plt.hist(p_diffs, label='p_diffreices')
         plt.xlabel('p difference')
         plt.ylabel('p differecne frequency');
```



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

```
In [32]: (p_diffs > obs_diffs).mean()
```

```
Out[32]: 0.83899999999999997
```

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?

Put your answer here.

-

0.3 NOTE null hypothesis { p_new - p_old <= 0 }

-

0.3.1 ANSWER

1. FAIL TO REJECT THE NULL
2. THE NEW OLD PAGE IS STILL FIT TO BE USED

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 [33]: import statsmodels.api as sm

         convert_old = (df2.query('group == "control"').query('converted == 1').converted).count
         convert_new = (df2.query('group == "treatment"').query('converted == 1').converted).cou

         n_old = df2.query('group == "control"').converted.count()
         n_new = df2.query('group == "treatment"').converted.count()

         convert_old, convert_new, n_old, n_new
```

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

```
Out[33]: (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 [34]: from statsmodels.stats.proportion import proportions_ztest

         count = np.array([convert_old, convert_new])
         nobs = np.array([n_old, n_new])

         stats, pval = proportions_ztest(count, nobs, alternative='smaller')

         print("z_score : {}, p_valus : {}".format(stats, pval))
```

```
z_score : 1.3109241984234394, p_valus : 0.9050583127590245
```

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.**?

Put your answer here.

Answer - Z_score 1.31092 show how any standard diveation is away from the p_vause 0.18988 - YES i do agree with me findding in rejecting the null hypothesis

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?

Put your answer here. ## - Logistic Regresion

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 [35]: df2['intercept'] = 1

         df2['ab_page'] = [1 if x == 'treatment' else 0 for x in df2['group'] ]

         df2.head(10)
```

Out[35]:

	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
5	936923	2017-01-10 15:20:49.083499	control	old_page	0	1	0
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1	1	1
7	719014	2017-01-17 01:48:29.539573	control	old_page	0	1	0
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1	1	1
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1	1	1

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 [36]: import statsmodels.api as sm

In [37]: # ,'old_page', 'new_page'
         logit_var = df2[['intercept', 'ab_page']]
         logit_mod = sm.Logit(df2['converted'], logit_var)
         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 [38]: results.summary2()

Out[38]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                 Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-06-26 12:33 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**? **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**?

Put your answer here. - P_value associated with ab_page = {0.1899} - part II we did the total number of successes against total number of events occuring to calculate the p_valuse which was a one sided test and also part 3 was a two sidded test which will lead to a larger picture of p-value

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?

Put your answer here. 1. adding a dummy variable for new_page and old page into the logistic regression might be a disadvantage in making the p_value statisticaly not significant

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.

```
In [39]: countries_df = pd.read_csv('./countries.csv')
         df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
         df_new.head()
```

Out[39]:

user_id	country	timestamp	group	landing_page	converted	intercept	ab_page
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 [40]: # Create the necessary dummy variables
         df_new = df_new.join(pd.get_dummies(df_new.country))

         logit_var = df_new[['intercept', 'CA', 'UK']]

         logit_mod = sm.Logit(df_new.converted, logit_var)

         results = logit_mod.fit()
         results.summary2()

Optimization terminated successfully.
         Current function value: 0.366116
         Iterations 6


Out[40]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
```

```
==========================================================================
Model:              Logit            No. Iterations:    6.0000
Dependent Variable: converted        Pseudo R-squared:  0.000
Date:               2020-06-26 12:33 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    -1.9967    0.0068  -292.3145  0.0000  -2.0101  -1.9833
CA           -0.0408    0.0269    -1.5178  0.1291  -0.0935   0.0119
UK            0.0099    0.0133     0.7458  0.4558  -0.0161   0.0360
==========================================================================

"""
```

```
In [41]: print('P_CA : {}, p_UK : {}'.format(1/np.exp(-0.0408), np.exp(0.0099)))
```

```
P_CA : 1.0416437559600236, p_UK : 1.0099491671175422
```

Question does it apper that county has an impact conversion - ANSWER - For every one unit decrease CA convertion rate is 1.0416 times likely to occure holding all else constant - For every one unit increas UK convertion rate is 1.0099 times likely to occure holding all else constant

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]: ### Fit Your Linear Model And Obtain the Results
         df_new['US_page'] = df_new['ab_page'] + df_new['US']
         df_new['CA_page'] = df_new['ab_page'] + df_new['CA']
         df_new['UK_page'] = df_new['ab_page'] + df_new['UK']
         logit_var = df_new[['intercept', 'US_page', 'CA_page', 'UK_page']]
         logit_mod = sm.Logit(df_new.converted, logit_var)
         results = logit_mod.fit()
         results.summary2()
```

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

```
Out[42]: <class 'statsmodels.iolib.summary2.Summary'>
         """
```

```
                          Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2020-06-26 12:33 AIC:              212781.1253
No. Observations:   290584           BIC:              212823.4439
Df Model:           3                Log-Likelihood:   -1.0639e+05
Df Residuals:       290580           LL-Null:          -1.0639e+05
Converged:          1.0000           Scale:            1.0000
-----------------------------------------------------------------
              Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
-----------------------------------------------------------------
intercept    -1.9946    0.0136  -146.6071  0.0000  -2.0212  -1.9679
US_page       0.0053    0.0112     0.4752  0.6346  -0.0166   0.0272
CA_page      -0.0354    0.0183    -1.9384  0.0526  -0.0713   0.0004
UK_page       0.0152    0.0124     1.2288  0.2191  -0.0090   0.0394
=================================================================

"""
```

```
In [43]: print('US_page : {}, CA_page: {}, UK_page: {}'.format(np.exp(0.0053), 1/np.exp(-0.0354)
```

```
US_page : 1.0053140698457452, CA_page: 1.0360340395437675, UK_page: 1.015316107532257
```

conculution - For Every One unit increase in US page the convertion rate will occure 1.0053 holding all variable constant - For Every One unit decrease in CA page the convertion rate will occure 1.0360 holding all variabls constant - For Every One unit increase in UK page the convertion rate will occure 1.0153 holding all varibale constant
- This implies base on the p_values which is greater than 0.05 an interaction between countries and pages have no effect on conversion

0.4 conclusions

- The old page is better than the new page

- Location has no effect on interaction between countries and pages for convertion
- More time time requerid becouse the old page had more occuring event than the new page which might be the couse of the old page being better than the new page

Conclusions
Congratulations on completing the project!

0.4.1 Gather Submission Materials

Once you are satisfied with the status of your Notebook, you should save it in a format that will make it easy for others to read. You can use the **File -> Download as -> HTML (.html)** menu to save your notebook as an .html file. If you are working locally and get an error about "No

module name", then open a terminal and try installing the missing module using `pip install <module_name>` (don't include the "<" or ">" or any words following a period in the module name).

You will submit both your original Notebook and an HTML or PDF copy of the Notebook for review. There is no need for you to include any data files with your submission. If you made reference to other websites, books, and other resources to help you in solving tasks in the project, make sure that you document them. It is recommended that you either add a "Resources" section in a Markdown cell at the end of the Notebook report, or you can include a `readme.txt` file documenting your sources.

0.4.2 Submit the Project

When you're ready, click on the "Submit Project" button to go to the project submission page. You can submit your files as a .zip archive or you can link to a GitHub repository containing your project files. If you go with GitHub, note that your submission will be a snapshot of the linked repository at time of submission. It is recommended that you keep each project in a separate repository to avoid any potential confusion: if a reviewer gets multiple folders representing multiple projects, there might be confusion regarding what project is to be evaluated.

It can take us up to a week to grade the project, but in most cases it is much faster. You will get an email once your submission has been reviewed. If you are having any problems submitting your project or wish to check on the status of your submission, please email us at dataanalyst-project@udacity.com. In the meantime, you should feel free to continue on with your learning journey by beginning the next module in the program.

In [ ]: