Notebook

February 3, 2024

1 A/B Testing Result Analysis

This dataset is the outcome of an A/B Testing Experiment, focusing on the introduction of a new landing page alongside the continued use of the old one

Objective: To analyze the **Conversion Rate** between two groups: - Control : Individuals exposed to the old landing page - Treatment: Individuals exposed to the new landing page

Expected Outcome:

The goal is to provide a recommendation on whether implementing the new landing page is advisable, with the aim of achieving a significant improvement in the **Conversion Rate**

[ ]:
```
import pandas as pd
from google.colab import files
ab_data = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving ab_data.csv to ab_data.csv
```

1. Data Cleaning

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

[ ]:

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

Examine the period of the experiment

[ ]:
```
print(df['timestamp'].min())
print(df['timestamp'].max())
```

```
2017-01-02 13:42:05.378582
2017-01-24 13:41:54.460509
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   user_id       294478 non-null  int64
 1   timestamp     294478 non-null  object
 2   group         294478 non-null  object
 3   landing_page  294478 non-null  object
 4   converted     294478 non-null  int64
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
```

Filter the rows containing wrong pair of 'group' and 'landing_page'

[ ]:
```
df = df[((df['group']=='control')&(df['landing_page']=='old_page') |␣
  ↪(df['group']=='treatment')&(df['landing_page']=='new_page'))]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290585 entries, 0 to 294477
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   user_id       290585 non-null  int64
 1   timestamp     290585 non-null  object
 2   group         290585 non-null  object
 3   landing_page  290585 non-null  object
 4   converted     290585 non-null  int64
dtypes: int64(2), object(3)
memory usage: 13.3+ MB
```

Check duplicates

[ ]:
```
df['user_id'].duplicated().sum()
```

[ ]:
```
1
```

[ ]:
```
df[df['user_id'].duplicated()]
```

[ ]:

	user_id	timestamp	group	landing_page	converted
2893	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

[ ]:
```
df[df['user_id']==773192]
```

[ ]:

	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

[ ]:
```
df['timestamp'] = df.groupby('user_id')['timestamp'].transform('min')
df = df.drop_duplicates()
df['user_id'].duplicated().sum()
```

[ ]: 0

Cross Tabulation for 'group' and 'converted'

[ ]:
```
pd.crosstab(df['group'], df['converted'])
```

[ ]:

converted	0	1
group		
control	127785	17489
treatment	128046	17264

The proportions of 'converted' values in the control and treatment groups are notably similar.

As per the report presented by the Marketing Department, the existing conversion rate stands at 12% **(Baseline Proportion = 0.12)**

The objective is to elevate the Conversion Rate to 14% **(Expected Proportion = 0.14).**

[ ]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import statsmodels.stats.api as sms
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from math import ceil
import os
```

2. Choose Sample Size: How many people should we have in each group?

[ ]:
```
# step 1: calculate the effect size: a measure of how practically significant␣
 ↪the difference is between the control and treatment groups
effect_size = sms.proportion_effectsize(0.12, 0.14) # the baseline proportion␣
 ↪and expected proportion

# step 2: calculate sample size (required_n)
required_n = sms.NormalIndPower().solve_power(
    effect_size,
    power = 0.8,
    alpha = 0.05,
    ratio = 1
```

```
)

required_n = ceil(required_n) # rounding up to next
required_n
```

[ ]: 4433

Subset the data according to required sample size

[ ]:
```
control_sample = df[df['group']=='control'].sample(n=required_n,␣
 ↪random_state=25)
treatment_sample = df[df['group']=='treatment'].sample(n=required_n,␣
 ↪random_state=25)

ab_data = pd.concat([control_sample, treatment_sample], axis=0)
ab_data.reset_index(drop=True, inplace=True)
ab_data
```

[ ]:

	user_id	timestamp	group	landing_page	converted
0	814711	2017-01-15 15:26:57.871076	control	old_page	0
1	647064	2017-01-15 10:08:24.495491	control	old_page	0
2	853275	2017-01-11 22:13:53.176145	control	old_page	0
3	866769	2017-01-11 19:28:25.182679	control	old_page	0
4	845104	2017-01-05 03:31:06.127408	control	old_page	0
…	…	…	…	…	…
8861	660355	2017-01-21 10:33:17.712420	treatment	new_page	0
8862	857451	2017-01-03 16:33:09.338872	treatment	new_page	0
8863	881093	2017-01-18 03:18:02.466992	treatment	new_page	0
8864	814800	2017-01-14 21:10:16.244511	treatment	new_page	0
8865	746931	2017-01-15 15:26:42.295806	treatment	new_page	0

[8866 rows x 5 columns]

[ ]:
```
ab_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8866 entries, 0 to 8865
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   user_id       8866 non-null   int64
 1   timestamp     8866 non-null   object
 2   group         8866 non-null   object
 3   landing_page  8866 non-null   object
 4   converted     8866 non-null   int64
dtypes: int64(2), object(3)
memory usage: 346.5+ KB
```

3. Calculate Conversion Rate of each group

[ ]:
```
ab_data.groupby('group')['converted'].agg(np.mean)
```

[ ]:
```
group
control      0.114370
treatment    0.124521
Name: converted, dtype: float64
```

4. Hypothesis Development

Significance Level (alpha) = 0.05

Confidence Level = 0.95

P0 = conversion rate of old page

P1 = conversion rate of new page

- **H0: P0 - P1 = 0**
- **HA: P0 - P1 <> 0**

[ ]:
```
from statsmodels.stats.proportion import proportions_ztest, proportion_confint
```

5. Compute essential metrics for Z-statistics

[ ]:
```
control_size = len(control_sample)
treatment_size = len(treatment_sample)
successes = [control_sample['converted'].sum(), treatment_sample['converted'].
 ↪sum()]
nobs = [control_size, treatment_size]

z_stats, p_value = proportions_ztest(successes, nobs=nobs)
(lwr_control, lwr_treatment), (upr_control, upr_treatment) =␣
 ↪proportion_confint(successes, nobs=nobs, alpha=0.05)

print(f'z_stats: {z_stats:.3f}')
print(f'p_value: {p_value:.3f}')
print(f'confidence interval 95% - control group: [{lwr_control:.3f},␣
 ↪{upr_control:.3f}]')
print(f'confidence interval 95% - treatment group: [{lwr_treatment:.3f},␣
 ↪{upr_treatment:.3f}]')
```

```
z_stats: -1.474
p_value: 0.141
confidence interval 95% - control group: [0.105, 0.124]
confidence interval 95% - treatment group: [0.115, 0.134]
```

Findings

When observing a p-value greater than 0.05

We accept the null hypothesis: P1 - P0 = 0

There is no discernible difference in conversion rates between the control and treatment groups

Confidence intervals for both groups largely overlap

Recommendation: It is advised not to implement the treatment (new landing page) as there appears to be no statistically significant improvement in conversion rates compared to the control group

[ ]:
```
!pip install nbconvert
!apt-get install texlive-xetex
```

[3]:
```
from google.colab import drive
import nbformat
from nbconvert import PDFExporter

# Mount Google Drive
drive.mount('/content/drive')

# Get the notebook name
notebook_name = 'AB_Test_Landing_Page.ipynb'

# Load the notebook
notebook_path = f'/content/drive/My Drive/Colab Notebooks/{notebook_name}'
with open(notebook_path) as f:
  notebook = nbformat.read(f, as_version=4)

# Configure PDF export
pdf_exporter = PDFExporter()
pdf_data, resources = pdf_exporter.from_notebook_node(notebook)

# Save PDF to Google Drive
pdf_path = f'/content/drive/My Drive/Colab Notebooks/{notebook_name.replace(".
 ↪ipynb", ".pdf")}'

with open(pdf_path, 'wb') as f:
  f.write(pdf_data)

print(f'PDF saved to: {pdf_path}')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
PDF saved to: /content/drive/My Drive/Colab Notebooks/AB_Test_Landing_Page.pdf
```