kxj5bmx3p

May 2, 2024

1 A/B Testing

Choosing a best marketing model is required for better sales and revenue.A/B Testing means choosing the best marketing strategy by analysing two marketing strategies.Suppose we are marketing our product on social media on one target users and will do the same marketing strategy in different user base.After analysing the results of both the campines, I will select the best campingn will gives better reach. Here our goal is to boost the sales,revenue,reach and followers. This will be done by comparing two campingns.This technique is called A/B Testing. For the Data we need to get the datasets of two different marketing strategies for the same goal.

[1]:
```
import pandas as pd
import datetime
from datetime import date, timedelta
import plotly.graph_objects as go
import plotly.express as px
import plotly.io as pio
pio.templates.default = "plotly_white"

control_data = pd.read_csv(r"C:\Masters\Forage\Certificates\RESUME\portfolio\AB␣
 ↪Testing\Datasets\control_group.csv", sep = ";")
test_data = pd.read_csv(r"C:\Masters\Forage\Certificates\RESUME\portfolio\AB␣
 ↪Testing\Datasets\test_group.csv", sep = ";")
```

[6]:
```
control_data.head()
```

[6]:

	Campaign Name	Date	Spend [USD]	# of Impressions	Reach	# of Website Clicks	# of Searches	# of View Content	# of Add to Cart
0	Control Campaign	1.08.2019	2280	82702.0	56930.0	7016.0	2290.0	2159.0	1819.0
1	Control Campaign	2.08.2019	1757	121040.0	102513.0	8110.0	2033.0	1841.0	1219.0
2	Control Campaign	3.08.2019	2343	131711.0	110862.0	6508.0	1737.0	1549.0	1134.0
3	Control Campaign	4.08.2019	1940	72878.0	61235.0	3065.0	1042.0	982.0	1183.0
4	Control Campaign	5.08.2019	1835	NaN	NaN	NaN	NaN	NaN	NaN

```
   # of Purchase
0          618.0
1          511.0
2          372.0
3          340.0
4            NaN
```

[7]:
```
test_data.head()
```

[7]:
```
   Campaign Name       Date  Spend [USD]  # of Impressions  Reach  \
0  Test Campaign  1.08.2019         3008             39550  35820
1  Test Campaign  2.08.2019         2542            100719  91236
2  Test Campaign  3.08.2019         2365             70263  45198
3  Test Campaign  4.08.2019         2710             78451  25937
4  Test Campaign  5.08.2019         2297            114295  95138

   # of Website Clicks  # of Searches  # of View Content  # of Add to Cart  \
0                 3038           1946               1069               894
1                 4657           2359               1548               879
2                 7885           2572               2367              1268
3                 4216           2216               1437               566
4                 5863           2106                858               956

   # of Purchase
0            255
1            677
2            578
3            340
4            768
```

2 Data Preprocessing

[8]:
```
control_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Campaign Name        30 non-null     object
 1   Date                 30 non-null     object
 2   Spend [USD]          30 non-null     int64
 3   # of Impressions     29 non-null     float64
 4   Reach                29 non-null     float64
 5   # of Website Clicks  29 non-null     float64
```

```
 6   # of Searches        29 non-null     float64
 7   # of View Content    29 non-null     float64
 8   # of Add to Cart     29 non-null     float64
 9   # of Purchase        29 non-null     float64
dtypes: float64(7), int64(1), object(2)
memory usage: 2.5+ KB
```

[9]:
```
control_data.describe()
```

[9]:

	Spend [USD]	# of Impressions	Reach	# of Website Clicks	\
count	30.000000	29.000000	29.000000	29.000000	
mean	2288.433333	109559.758621	88844.931034	5320.793103	
std	367.334451	21688.922908	21832.349595	1757.369003	
min	1757.000000	71274.000000	42859.000000	2277.000000	
25%	1945.500000	92029.000000	74192.000000	4085.000000	
50%	2299.500000	113430.000000	91579.000000	5224.000000	
75%	2532.000000	121332.000000	102479.000000	6628.000000	
max	3083.000000	145248.000000	127852.000000	8137.000000	

	# of Searches	# of View Content	# of Add to Cart	# of Purchase
count	29.000000	29.000000	29.000000	29.000000
mean	2221.310345	1943.793103	1300.000000	522.793103
std	866.089368	777.545469	407.457973	185.028642
min	1001.000000	848.000000	442.000000	222.000000
25%	1615.000000	1249.000000	930.000000	372.000000
50%	2390.000000	1984.000000	1339.000000	501.000000
75%	2711.000000	2421.000000	1641.000000	670.000000
max	4891.000000	4219.000000	1913.000000	800.000000

[12]:
```
#checking any null values present in the control_data dataset
control_data.isnull().sum()
```

[12]:
```
Campaign Name          0
Date                   0
Spend [USD]            0
# of Impressions       1
Reach                  1
# of Website Clicks    1
# of Searches          1
# of View Content      1
# of Add to Cart       1
# of Purchase          1
dtype: int64
```

The control_data dataset contains null values on multiple columns. We can also see that most of the columns have hashtag/special character on it's column name.For analysis and ease of understanding we will change the column names and then will handle missing columns.

[13]:
```
#Changing column names to meaningful names
control_data.columns = ["Campaign Name", "Date", "Amount Spent",
                        "Number of Impressions", "Reach", "Website Clicks",
                        "Searches Received", "Content Viewed", "Added to Cart",
                        "Purchases"]
```

[21]:
```
control_data.head()
```

[21]:

	Campaign Name	Date	Amount Spent	Number of Impressions	Reach	Website Clicks	Searches Received	Content Viewed	Added to Cart	Purchases
0	Control Campaign	1.08.2019	2280	82702.000000	56930.000000	7016.000000	2290.000000	2159.000000	1819.0	618.000000
1	Control Campaign	2.08.2019	1757	121040.000000	102513.000000	8110.000000	2033.000000	1841.000000	1219.0	511.000000
2	Control Campaign	3.08.2019	2343	131711.000000	110862.000000	6508.000000	1737.000000	1549.000000	1134.0	372.000000
3	Control Campaign	4.08.2019	1940	72878.000000	61235.000000	3065.000000	1042.000000	982.000000	1183.0	340.000000
4	Control Campaign	5.08.2019	1835	109559.758621	88844.931034	5320.793103	2221.310345	1943.793103	1300.0	522.793103

[15]:
```
#Handling missing values
control_data["Number of Impressions"].fillna(value=control_data["Number of␣
 ↪Impressions"].mean(),
                                             inplace=True)
control_data["Reach"].fillna(value=control_data["Reach"].mean(),
                             inplace=True)
control_data["Website Clicks"].fillna(value=control_data["Website Clicks"].
 ↪mean(),
                                      inplace=True)
control_data["Searches Received"].fillna(value=control_data["Searches␣
 ↪Received"].mean(),
                                         inplace=True)
control_data["Content Viewed"].fillna(value=control_data["Content Viewed"].
 ↪mean(),
                                      inplace=True)
control_data["Added to Cart"].fillna(value=control_data["Added to Cart"].mean(),
                                     inplace=True)
control_data["Purchases"].fillna(value=control_data["Purchases"].mean(),
```

```
                                        inplace=True)
```

Similarly for test_data, we will change the coulumn names first and then handle the missing data.

[37]:
```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Campaign Name          30 non-null     object
 1   Date                   30 non-null     object
 2   Amount Spent           30 non-null     int64
 3   Number of Impressions  30 non-null     int64
 4   Reach                  30 non-null     int64
 5   Website Clicks         30 non-null     int64
 6   Searches Received      30 non-null     int64
 7   Content Viewed         30 non-null     int64
 8   Added to Cart          30 non-null     int64
 9   Purchases              30 non-null     int64
dtypes: int64(8), object(2)
memory usage: 2.5+ KB
```

[38]:
```
test_data.describe()
```

[38]:

	Amount Spent	Number of Impressions	Reach	Website Clicks	\
count	30.000000	30.000000	30.000000	30.000000	
mean	2563.066667	74584.800000	53491.566667	6032.333333	
std	348.687681	32121.377422	28795.775752	1708.567263	
min	1968.000000	22521.000000	10598.000000	3038.000000	
25%	2324.500000	47541.250000	31516.250000	4407.000000	
50%	2584.000000	68853.500000	44219.500000	6242.500000	
75%	2836.250000	99500.000000	78778.750000	7604.750000	
max	3112.000000	133771.000000	109834.000000	8264.000000	

	Searches Received	Content Viewed	Added to Cart	Purchases
count	30.000000	30.000000	30.000000	30.000000
mean	2418.966667	1858.000000	881.533333	521.233333
std	388.742312	597.654669	347.584248	211.047745
min	1854.000000	858.000000	278.000000	238.000000
25%	2043.000000	1320.000000	582.500000	298.000000
50%	2395.500000	1881.000000	974.000000	500.000000
75%	2801.250000	2412.000000	1148.500000	701.000000
max	2978.000000	2801.000000	1391.000000	890.000000

[17]:
```
#Checking for missing values on test_data dataset
test_data.isnull().sum()
```

[17]:
```
Campaign Name          0
Date                   0
Spend [USD]            0
# of Impressions       0
Reach                  0
# of Website Clicks    0
# of Searches          0
# of View Content      0
# of Add to Cart       0
# of Purchase          0
dtype: int64
```

We can see that there aren't any null values in the dataset and it is pretty clean.Now will handle the column names.

[18]:
```
#Giving the new meaningful names to the test_data dataset.
test_data.columns = ["Campaign Name", "Date", "Amount Spent",
                     "Number of Impressions", "Reach", "Website Clicks",
                     "Searches Received", "Content Viewed", "Added to Cart",
                     "Purchases"]
```

[20]:
```
test_data.head()
```

[20]:

	Campaign Name	Date	Amount Spent	Number of Impressions	Reach	\
0	Test Campaign	1.08.2019	3008	39550	35820	
1	Test Campaign	2.08.2019	2542	100719	91236	
2	Test Campaign	3.08.2019	2365	70263	45198	
3	Test Campaign	4.08.2019	2710	78451	25937	
4	Test Campaign	5.08.2019	2297	114295	95138	

	Website Clicks	Searches Received	Content Viewed	Added to Cart	Purchases
0	3038	1946	1069	894	255
1	4657	2359	1548	879	677
2	7885	2572	2367	1268	578
3	4216	2216	1437	566	340
4	5863	2106	858	956	768

Now we will merge the both control_data and test_data for analysis

[23]:
```
abtesting_data = control_data.merge(test_data,
                                    how="outer").sort_values(["Date"])
abtesting_data = abtesting_data.reset_index(drop=True)
abtesting_data.head()
```

```
C:\Users\16052\AppData\Local\Temp\ipykernel_25452\2267744386.py:1: UserWarning:
```

```
You are merging on int and float columns where the float values are not equal to
their int representation.
  abtesting_data = control_data.merge(test_data,
```

[23]:

	Campaign Name	Date	Amount Spent	Number of Impressions	Reach	\
0	Control Campaign	1.08.2019	2280	82702.0	56930.0	
1	Test Campaign	1.08.2019	3008	39550.0	35820.0	
2	Test Campaign	10.08.2019	2790	95054.0	79632.0	
3	Control Campaign	10.08.2019	2149	117624.0	91257.0	
4	Test Campaign	11.08.2019	2420	83633.0	71286.0	

	Website Clicks	Searches Received	Content Viewed	Added to Cart	Purchases
0	7016.0	2290.0	2159.0	1819.0	618.0
1	3038.0	1946.0	1069.0	894.0	255.0
2	8125.0	2312.0	1804.0	424.0	275.0
3	2277.0	2475.0	1984.0	1629.0	734.0
4	3750.0	2893.0	2617.0	1075.0	668.0

3 AB Testing

Firstly we will check for the relationship between number of impressions we got and the amount spent from both the campigns.

[26]:
```
figure = px.scatter(data_frame = abtesting_data,
                    x="Number of Impressions",
                    y="Amount Spent",
                    size="Amount Spent",
                    color= "Campaign Name",
                    trendline="ols")
figure.show()
```

By seeing the above plot we can clearly say that the Control campign's number of impressions are directly proportional to the Amount spent of the campaign.In the next plat we will see number of cummulative searches on each campaign.

[27]:
```
label = ["Total Searches from Control Campaign",
         "Total Searches from Test Campaign"]
counts = [sum(control_data["Searches Received"]),
          sum(test_data["Searches Received"])]
colors = ['gold','lightgreen']
fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Control Vs Test: Searches')
fig.update_traces(hoverinfo='label+percent', textinfo='value',
                  textfont_size=30,
                  marker=dict(colors=colors,
                              line=dict(color='black', width=3)))
fig.show()
```

By seeing the above pie chart we can say that test campaign have more searches.now we will see number of website clicks from both the campaigns.

[28]:
```
label = ["Website Clicks from Control Campaign",
         "Website Clicks from Test Campaign"]
counts = [sum(control_data["Website Clicks"]),
          sum(test_data["Website Clicks"])]
colors = ['gold','lightgreen']
fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Control Vs Test: Website Clicks')
fig.update_traces(hoverinfo='label+percent', textinfo='value',
                  textfont_size=30,
                  marker=dict(colors=colors,
                              line=dict(color='black', width=3)))
fig.show()
```

Test campaign have more number of website clicks that the control campaign.Now we will see the amount of content viewed after reaching the website from both campaigns

[29]:
```
label = ["Content Viewed from Control Campaign",
         "Content Viewed from Test Campaign"]
counts = [sum(control_data["Content Viewed"]),
          sum(test_data["Content Viewed"])]
colors = ['gold','lightgreen']
fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Control Vs Test: Content Viewed')
fig.update_traces(hoverinfo='label+percent', textinfo='value',
                  textfont_size=30,
                  marker=dict(colors=colors,
                              line=dict(color='black', width=3)))
fig.show()
```

Here we can see that the more users watched the control campaign but there is no big difference in numbers.In the following code we will see number of products added to the cart.

[30]:
```
label = ["Products Added to Cart from Control Campaign",
         "Products Added to Cart from Test Campaign"]
counts = [sum(control_data["Added to Cart"]),
          sum(test_data["Added to Cart"])]
colors = ['gold','lightgreen']
fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Control Vs Test: Added to Cart')
fig.update_traces(hoverinfo='label+percent', textinfo='value',
                  textfont_size=30,
                  marker=dict(colors=colors,
                              line=dict(color='black', width=3)))
fig.show()
```

since it has low website clicks more products were added to the cart from the control campaign. Now we will see amount spent on both campaigns

[31]:
```
label = ["Amount Spent in Control Campaign",
         "Amount Spent in Test Campaign"]
counts = [sum(control_data["Amount Spent"]),
          sum(test_data["Amount Spent"])]
colors = ['gold','lightgreen']
fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Control Vs Test: Amount Spent')
fig.update_traces(hoverinfo='label+percent', textinfo='value',
                  textfont_size=30,
                  marker=dict(colors=colors,
                              line=dict(color='black', width=3)))
fig.show()
```

we can see that the amount spent of test campaign is higher that the amount spent on control campaign.But as we can see that the control campaign makes more in content views and more products in the cart, the control campaign is more efficient than the test campaign. Now we will see the purchases made by both campaigns.

[32]:
```
label = ["Purchases Made by Control Campaign",
         "Purchases Made by Test Campaign"]
counts = [sum(control_data["Purchases"]),
          sum(test_data["Purchases"])]
colors = ['gold','lightgreen']
fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Control Vs Test: Purchases')
fig.update_traces(hoverinfo='label+percent', textinfo='value',
                  textfont_size=30,
                  marker=dict(colors=colors,
                              line=dict(color='black', width=3)))
fig.show()
```

The purchases made by both campaigns are almost same.But control campaign makes more sales in less amount spent on marketing.I belive some users are very particular, they don't even spend even 3 seconds,if they don't like the content.Now will will see the relation between content added to the cart and content viewed.

[34]:
```
figure = px.scatter(data_frame = abtesting_data,
                    x="Added to Cart",
                    y="Content Viewed",
                    size="Added to Cart",
                    color= "Campaign Name",
                    trendline="ols")
figure.show()
```

As we can see that the control campaign wins here.NOw will draw a realtionship between number of sales and number of products added to the cart.

[36]:
```
figure = px.scatter(data_frame = abtesting_data,
                    x="Purchases",
                    y="Added to Cart",
                    size="Purchases",
                    color= "Campaign Name",
                    trendline="ols")
figure.show()
```

Here we can see the linear growth in the test campaign.

4 Conclusion:

From the A/B Testing control campaign makes more sales and user engagement.Based on the analysis , It is clear that more number of users viewed control campaign, they added more products to their cart and this results in more sales.But the Test campaign results in more sales overall as per the products viewed and added to the cart, But the control campaign makes more sales overall. Hence we can use control campaign to market multiple products to multiple users, And Test campaign can be used to market single product to specific user grop.