Notebook

February 19, 2025

[275]:
```
import csv
import pandas as pd
import numpy as np
```

[277]:
```
merged_df = pd.read_csv('chips_merged.csv')
print(merged_df.sort_values('Transaction_ID'))
```

```
              Date  Store_Number  Loyalty_Card_Number  Transaction_ID  \
0       2018-10-17             1                 1000               1
240664  2018-09-16             1                 1002               2
188931  2019-03-07             1                 1003               3
188932  2019-03-08             1                 1003               4
102787  2018-11-02             1                 1004               5
…              …             …                    …               …
25107   2018-09-26           272               272392          270206
25108   2018-09-29           272               272392          270207
25109   2018-10-31           272               272392          270208
25110   2019-02-17           272               272392          270209
15829   2018-12-20            88               237324         2415841

        Product_Number                            Product_Name  \
0                    5      Natural Chip Compny SeaSalt175g
240664              58  Red Rock Deli Chikn&Garlic Aioli 150g
188931              52    Grain Waves Sour Cream&Chives 210G
188932             106     Natural ChipCo Hony Soy Chckn175g
102787              96        WW Original Stacked Chips 160g
…                  …                                      …
25107               70    Tyrrells Crisps Lightly Salted 165g
25108               75          Cobs Popd Sea Salt Chips 110g
25109               81          Pringles Original Crisps 134g
25110               78       Thins Chips Salt & Vinegar 175g
15829              102   Kettle Mozzarella Basil & Pesto 175g

        Product_Quantity  Total_Sales              Life_Stage Customer_Type
0                      2          6.0  Young singles/couples       Premium
240664                 1          2.7  Young singles/couples    Mainstream
188931                 1          3.6          Young families        Budget
188932                 1          3.0          Young families        Budget
```

```
102787                  1          1.9   Older singles/couples    Mainstream
…                    …          …                    …           …
25107                   2          8.4  Midage singles/couples       Premium
25108                   2          7.6  Midage singles/couples       Premium
25109                   2          7.4  Midage singles/couples       Premium
25110                   2          6.6  Midage singles/couples       Premium
15829                   2         10.8  Midage singles/couples    Mainstream

[264835 rows x 10 columns]
```

[279]:
```
control_df = pd.read_csv('chips_merged.csv')

control_df['Date'] = pd.to_datetime(control_df['Date'])

control_df['Numbered_Life_Stage'] = control_df['Life_Stage'].astype('category').
  ↪cat.codes
control_df['Numbered_Customer_Type'] = control_df['Customer_Type'].
  ↪astype('category').cat.codes

control_start, control_end = "2018-07-01", "2019-01-31"
control_dataframe = control_df[(control_df['Date'] >= control_start) &␣
  ↪(control_df['Date'] <= control_end)]

num_months = 7

control_aggregated = control_dataframe.groupby('Store_Number').agg(
    Total_Sales=('Total_Sales', 'sum'),
    Total_Sales_Average=('Total_Sales', 'sum'),
    Number_of_Transactions=('Transaction_ID', 'count'),
    Average_Number_of_Transactions=('Transaction_ID', 'count'),
    Number_of_Unique_Customers=('Loyalty_Card_Number', 'nunique'),
    Average_Number_of_Unique_Customers=('Loyalty_Card_Number', 'nunique'),
    Average_Items_per_Transaction=('Product_Quantity', 'mean'),
    Average_Life_Stage=('Numbered_Life_Stage', 'mean'),
    Average_Customer_Type=('Numbered_Customer_Type', 'mean')
).reset_index()

control_aggregated['Total_Sales_Average'] /= num_months
control_aggregated['Average_Number_of_Transactions'] /= num_months
control_aggregated['Average_Number_of_Unique_Customers'] /= num_months

control_aggregated = control_aggregated[~control_aggregated['Store_Number'].
  ↪isin([31, 11])]

control_aggregated.sort_values(by='Store_Number', inplace=True)

control_aggregated.dropna(inplace=True)
```

```
[281]: trial_stores = [77, 86, 88]
       non_trial_stores = control_aggregated[~control_aggregated['Store_Number'].
        ↪isin(trial_stores)]['Store_Number'].unique()

       metrics = ['Total_Sales', 'Total_Sales_Average', 'Number_of_Transactions',␣
        ↪'Average_Number_of_Transactions', 'Number_of_Unique_Customers',
                   'Average_Number_of_Unique_Customers',
                  'Average_Items_per_Transaction',
                  'Average_Life_Stage', 'Average_Customer_Type']
```

```
[283]: from scipy.spatial.distance import euclidean

       def find_best_control_distance(trial_store):
           best_store = None
           best_distance = float('inf')

           trial_data = control_aggregated[control_aggregated['Store_Number'] ==␣
        ↪trial_store].iloc[0, 1:]  # Get the row as Series

           for store in non_trial_stores:
               control_data = control_aggregated[control_aggregated['Store_Number'] ==␣
        ↪store].iloc[0, 1:]

               if not control_data.empty:
                   distance = euclidean(trial_data, control_data)
                   if distance < best_distance:
                       best_distance = distance
                       best_store = store

           return best_store, best_distance

       control_stores_distance = {trial_store: find_best_control_distance(trial_store)␣
        ↪for trial_store in trial_stores}

       for trial, (control, distance) in control_stores_distance.items():
           print(f"Trial Store {trial} -> Best Control Store: {control} (Distance:␣
        ↪{distance:.4f})")
```

```
Trial Store 77 -> Best Control Store: 188 (Distance: 35.2295)
Trial Store 86 -> Best Control Store: 13 (Distance: 16.2734)
Trial Store 88 -> Best Control Store: 237 (Distance: 21.7403)
```

```
[285]: # Creating the dataframe containing only data for the trial period.

       trial_df = pd.read_csv('chips_merged.csv')

       trial_df['Date'] = pd.to_datetime(trial_df['Date'])
```

```
trial_df['Numbered_Life_Stage'] = trial_df['Life_Stage'].astype('category').cat.
↪codes
trial_df['Numbered_Customer_Type'] = trial_df['Customer_Type'].
↪astype('category').cat.codes

trial_start, trial_end = "2019-02-01", "2019-04-30"
trial_dataframe = trial_df[(trial_df['Date'] >= trial_start) &␣
↪(trial_df['Date'] <= trial_end)]

num_months = 3

trial_aggregated = trial_dataframe.groupby('Store_Number').agg(
    Total_Sales=('Total_Sales', 'sum'),
    Total_Sales_Average=('Total_Sales', 'sum'),
    Number_of_Transactions=('Transaction_ID', 'count'),
    Average_Number_of_Transactions=('Transaction_ID', 'count'),
    Number_of_Unique_Customers=('Loyalty_Card_Number', 'nunique'),
    Average_Number_of_Unique_Customers=('Loyalty_Card_Number', 'nunique'),
    Average_Items_per_Transaction=('Product_Quantity', 'mean'),
    Average_Life_Stage=('Numbered_Life_Stage', 'mean'),
    Average_Customer_Type=('Numbered_Customer_Type', 'mean')
).reset_index()

trial_aggregated['Total_Sales_Average'] /= num_months
trial_aggregated['Average_Number_of_Transactions'] /= num_months
trial_aggregated['Average_Number_of_Unique_Customers'] /= num_months

trial_aggregated = trial_aggregated[~trial_aggregated['Store_Number'].isin([31,␣
↪11])]

trial_aggregated.sort_values(by='Store_Number', inplace=True)

trial_aggregated.dropna(inplace=True)
```

[287]:
```
metrics = ['Total_Sales', 'Total_Sales_Average', 'Number_of_Transactions',␣
↪'Average_Number_of_Transactions', 'Number_of_Unique_Customers',
           'Average_Number_of_Unique_Customers',
           'Average_Items_per_Transaction',
           'Average_Life_Stage', 'Average_Customer_Type']
```

[289]:
```
# I created this function to compare the performance of metrics between a trial␣
↪store and a control store during the trial period

def trial_comparison(trial_store, control_store):
    x = trial_aggregated[(trial_aggregated['Store_Number'] == trial_store) |␣
↪(trial_aggregated['Store_Number'] == control_store)]
```

```
    results = {}
    for metric in metrics:
            results[metric] = x.groupby('Store_Number')[metric].mean()
    return pd.DataFrame(results)

print(trial_comparison(77,81))
```

```
              Total_Sales  Total_Sales_Average  Number_of_Transactions  \
Store_Number
77                  777.0                259.0                   148.0
81                 3597.9               1199.3                   406.0

              Average_Number_of_Transactions  Number_of_Unique_Customers  \
Store_Number
77                                 49.333333                       124.0
81                                135.333333                       246.0

              Average_Number_of_Unique_Customers  \
Store_Number
77                                     41.333333
81                                     82.000000

              Average_Items_per_Transaction  Average_Life_Stage  \
Store_Number
77                                 1.581081            3.628378
81                                 1.980296            3.214286

              Average_Customer_Type
Store_Number
77                         0.925676
81                         0.876847
```

[291]:
```
# I created this function to compare the performance of metrics between a trial␣
 ↪store and a control store during the control period

def control_comparison(trial_store, control_store):
    x = control_aggregated[(control_aggregated['Store_Number'] == trial_store)␣
 ↪| (control_aggregated['Store_Number'] == control_store)]
    results = {}
    for metric in metrics:
            results[metric] = x.groupby('Store_Number')[metric].mean()
    return pd.DataFrame(results)
print(control_comparison(77,81))
```

```
              Total_Sales  Total_Sales_Average  Number_of_Transactions  \
Store_Number
77                 1699.0           242.714286                   317.0
81                 8260.3          1180.042857                   954.0
```

```
              Average_Number_of_Transactions  Number_of_Unique_Customers  \
Store_Number
77                                 45.285714                       239.0
81                                136.285714                       356.0

              Average_Number_of_Unique_Customers  \
Store_Number
77                                     34.142857
81                                     50.857143

              Average_Items_per_Transaction  Average_Life_Stage  \
Store_Number
77                                 1.526814            3.839117
81                                 1.964361            3.335430

              Average_Customer_Type
Store_Number
77                         0.911672
81                         0.860587
```

[293]:
```
# Finally, this function takes the results of the previous two functions and␣
↪calculates a percentage

def trial_change(trial_store, control_store):
    for metric in metrics:
        x = (trial_comparison(trial_store, control_store) /␣
↪control_comparison(trial_store, control_store))*100
    return x

print(trial_change(77, 188))
```

```
              Total_Sales  Total_Sales_Average  Number_of_Transactions  \
Store_Number
77              45.732784           106.709829               46.687697
188             53.757054           125.433125               50.511945

              Average_Number_of_Transactions  Number_of_Unique_Customers  \
Store_Number
77                                108.937960                   51.882845
188                               117.861206                   57.990868

              Average_Number_of_Unique_Customers  \
Store_Number
77                                    121.059972
188                                   135.312024

              Average_Items_per_Transaction  Average_Life_Stage  \
Store_Number
77                              103.554277          94.510760
188                             103.220124          92.937312

              Average_Customer_Type
Store_Number
77                       101.536052
188                      104.589495
```

[295]:
```
print(trial_change(86, 13))
```

```
              Total_Sales  Total_Sales_Average  Number_of_Transactions  \
Store_Number
13              47.884606           111.730747               46.927374
86              45.559940           106.306527               46.258503

              Average_Number_of_Transactions  Number_of_Unique_Customers  \
Store_Number
13                                109.497207                   77.642276
86                                107.936508                   84.645669

              Average_Number_of_Unique_Customers  \
Store_Number
13                                    181.165312
86                                    197.506562

              Average_Items_per_Transaction  Average_Life_Stage  \
Store_Number
13                               100.336323           99.335961
86                               100.446878           99.981618

              Average_Customer_Type
Store_Number
13                       101.948441
86                        91.963184
```

[297]:
```
print(trial_change(88, 237))
```

```
              Total_Sales  Total_Sales_Average  Number_of_Transactions  \
Store_Number
88              45.683959           106.595905               44.916821
237             40.747145            95.076671               40.074557

              Average_Number_of_Transactions  Number_of_Unique_Customers  \
Store_Number
88                                104.805915                   69.786096
237                                93.507300                   72.576177

              Average_Number_of_Unique_Customers  \
Store_Number
88                                      162.834225
237                                     169.344414

              Average_Items_per_Transaction  Average_Life_Stage  \
Store_Number
88                                   100.791803           97.696626
237                                  101.178689           99.649470

              Average_Customer_Type
Store_Number
88                        100.915495
237                        86.752362
```

```
[ ]:
```

This notebook was converted with convert.ploomber.io