Optimal Social Media Spend Strategy using Bayesian MMM Model

April 28, 2024

[1]:
```
#!pip install pymc arviz
```

[2]:
```
import pymc as pm
import arviz as az
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[7]:
```
data = pd.DataFrame({
    'Instagram': np.random.uniform(50, 200, 100),
    'Facebook': np.random.uniform(50, 200, 100),
    'Twitter': np.random.uniform(50, 200, 100),
    'TikTok': np.random.uniform(50, 200, 100),
    'ContentType': np.random.choice(['Video', 'Image', 'Text'], 100),
    'TargetGroup': np.random.choice(['Teen', 'Adult', 'Senior'], 100),
    'Sales': np.random.uniform(100, 500, 100),
})
```

[8]:
```
data.head(10)
```

[8]:
```
    Instagram    Facebook     Twitter      TikTok ContentType TargetGroup  \
0  186.114939  137.348440  111.687010  123.422873       Image       Adult
1  120.432813  174.706537  144.450466  169.807256       Video      Senior
2   81.381999  175.391604   78.088914  128.870421        Text      Senior
3  199.751569   95.228723   83.814072   70.056636       Image       Adult
4  100.625929  193.804476  189.343537  199.233688       Image      Senior
5  174.458977  199.211201   87.091471  135.056058       Image       Adult
6  177.607702  105.942897   69.105913  107.598955       Video        Teen
7  146.150201  182.837537  173.072092  114.858861       Video        Teen
8  138.686616  132.795946  190.119788  131.834190       Image      Senior
9  174.521381  156.492578   96.105478   75.846485       Image       Adult

        Sales
0  262.730006
1  254.324300
2  201.786481
3  459.756648
4  268.269815
```

```
5   293.960745
6   429.694962
7   323.282674
8   267.872527
9   435.138026
```

[9]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Instagram    100 non-null    float64
 1   Facebook     100 non-null    float64
 2   Twitter      100 non-null    float64
 3   TikTok       100 non-null    float64
 4   ContentType  100 non-null    object
 5   TargetGroup  100 non-null    object
 6   Sales        100 non-null    float64
dtypes: float64(5), object(2)
memory usage: 5.6+ KB
```

[10]:
```python
with pm.Model() as marketing_mix_model:
    beta_instagram = pm.Normal("beta_instagram", mu=0, sigma=10)
    beta_facebook = pm.Normal("beta_facebook", mu=0, sigma=10)
    beta_twitter = pm.Normal("beta_twitter", mu=0, sigma=10)
    beta_tiktok = pm.Normal("beta_tiktok", mu=0, sigma=10)

    intercept = pm.Normal("intercept", mu=0, sigma=10)
    sigma = pm.HalfNormal("sigma", sigma=10)

    # Linear model to predict sales based on marketing spend
    mu = (
        intercept +
        beta_instagram * data['Instagram'] +
        beta_facebook * data['Facebook'] +
        beta_twitter * data['Twitter'] +
        beta_tiktok * data['TikTok']
    )

    # Likelihood for observed data
    sales = pm.Normal("sales", mu=mu, sigma=sigma, observed=data['Sales'])

    # Sample the posterior
    trace = pm.sample(2000, tune=1000, cores=1, return_inferencedata=True)
```

```
Auto-assigning NUTS sampler…
```

```
Initializing NUTS using jitter+adapt_diag…
Sequential sampling (2 chains in 1 job)
NUTS: [beta_instagram, beta_facebook, beta_twitter, beta_tiktok, intercept,
sigma]

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Sampling 2 chains for 1_000 tune and 2_000 draw iterations (2_000 + 4_000 draws
total) took 20 seconds.
We recommend running at least 4 chains for robust computation of convergence
diagnostics
```

[ ]:

```
#!pip install fastapi uvicorn
```

[11]:

```
# Function to calculate optimal marketing spend based on Bayesian coefficients
def calculate_optimal_spend(trace, budget, content_type, target_group):
    # Extract posterior means for each coefficient
    beta_instagram = np.mean(az.extract_dataset(trace)["beta_instagram"])
    beta_facebook = np.mean(az.extract_dataset(trace)["beta_facebook"])
    beta_twitter = np.mean(az.extract_dataset(trace)["beta_twitter"])
    beta_tiktok = np.mean(az.extract_dataset(trace)["beta_tiktok"])

    # Calculate initial proportions for each channel
    total_coeff = beta_instagram + beta_facebook + beta_twitter + beta_tiktok
    instagram_proportion = beta_instagram / total_coeff
    facebook_proportion = beta_facebook / total_coeff
    twitter_proportion = beta_twitter / total_coeff
    tiktok_proportion = beta_tiktok / total_coeff

    # Default optimal spend based on budget and proportions
    optimal_spend = {
        "Instagram": budget * instagram_proportion,
        "Facebook": budget * facebook_proportion,
        "Twitter": budget * twitter_proportion,
        "TikTok": budget * tiktok_proportion,
    }

    # Adjustments based on content type
    if content_type == 'Video':
        # Favor channels known for video content
        optimal_spend["Instagram"] *= 1.2
        optimal_spend["TikTok"] *= 1.2

    if content_type == 'Image':
```

```
        # Favor Instagram and Facebook for image-based content
        optimal_spend["Instagram"] *= 1.1
        optimal_spend["Facebook"] *= 1.1

    if content_type == 'Text':
        # Favor Facebook and Twitter for text-based content
        optimal_spend["Facebook"] *= 1.1
        optimal_spend["Twitter"] *= 1.1

    # Adjustments based on target group
    if target_group == 'Teen':
        optimal_spend["Instagram"] *= 1.1
        optimal_spend["TikTok"] *= 1.2

    if target_group == 'Adult':
        optimal_spend["Facebook"] *= 1.1
        optimal_spend["Twitter"] *= 1.1

    if target_group == 'Senior':
        optimal_spend["Facebook"] *= 1.2
        optimal_spend["Twitter"] *= 1.2

    # Normalize to ensure budget constraint
    total_spend = sum(optimal_spend.values())
    if total_spend > budget:
        adjustment_factor = budget / total_spend
        for channel in optimal_spend:
            optimal_spend[channel] *= adjustment_factor

    return optimal_spend
```

[12]:
```
# Example input: given budget, content type, and target group
total_budget = 1000
content_type = "Video"
target_group = "Teen"

# Calculate optimal spend with updated marketing channels and given budget,␣
 ↪content type, and target group
optimal_spend = calculate_optimal_spend(trace, total_budget, content_type,␣
 ↪target_group)

# Display the optimal spend for each marketing channel
print("Optimal marketing spend allocation:")
for channel, spend in optimal_spend.items():
    print(f"{channel}: ${spend:.2f}")
```

```
Optimal marketing spend allocation:
Instagram: $420.27
```

```
Facebook: $82.98
Twitter: $239.12
TikTok: $257.64
```

```
/var/folders/rs/lghzr3gd4bd05ypml1jl8dlc0000gn/T/ipykernel_14547/1276860184.py:4
: FutureWarning: extract_dataset has been deprecated, please use extract
  beta_instagram = np.mean(az.extract_dataset(trace)["beta_instagram"])
/var/folders/rs/lghzr3gd4bd05ypml1jl8dlc0000gn/T/ipykernel_14547/1276860184.py:5
: FutureWarning: extract_dataset has been deprecated, please use extract
  beta_facebook = np.mean(az.extract_dataset(trace)["beta_facebook"])
/var/folders/rs/lghzr3gd4bd05ypml1jl8dlc0000gn/T/ipykernel_14547/1276860184.py:6
: FutureWarning: extract_dataset has been deprecated, please use extract
  beta_twitter = np.mean(az.extract_dataset(trace)["beta_twitter"])
/var/folders/rs/lghzr3gd4bd05ypml1jl8dlc0000gn/T/ipykernel_14547/1276860184.py:7
: FutureWarning: extract_dataset has been deprecated, please use extract
  beta_tiktok = np.mean(az.extract_dataset(trace)["beta_tiktok"])
```

[ ]:

```

```