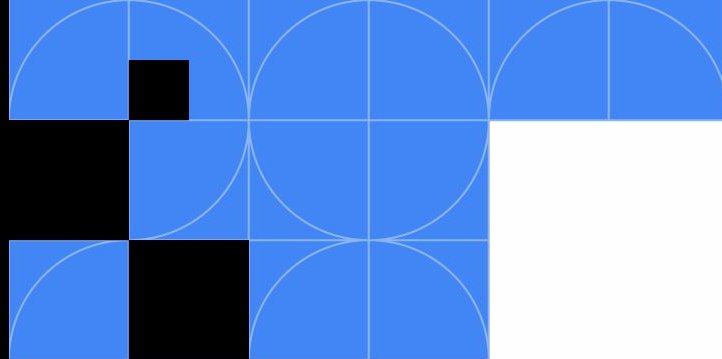


Build with AI



Creating Marketing models with AI

권정민 (ML GDE)



Google Developer Groups

United



발표자: **권정민**

- 프리랜서 데이터 분석가
- ODK Media, 우아한형제들, SK플래닛 등 데이터 분석
- [데이터를 엮는 사람들], [데이터 분석가의 숫자유감] 집필, 데이터 분석 관련 기술서 번역 및 감수
- AI/ML GDE, Women TechMaker Ambassador

Table of Contents

01 The GenAI Era (feat.Gemini)



02 Marketing Mix Model



03 Marketing Mix Model + Gemini Use Case



04 Q&A

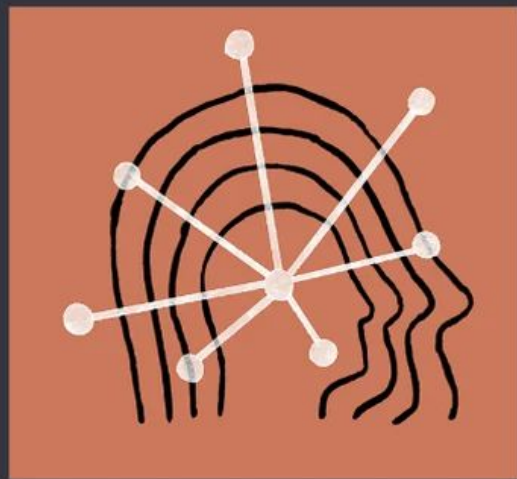


Build with AI



Google Developer Groups
United

The GenAI Era (feat. Gemini)



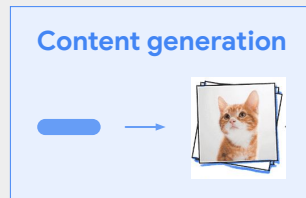
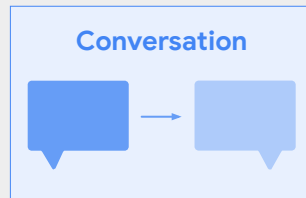
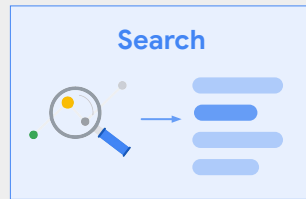
Gemini vs ChatGPT vs Claude

LLM이 주목받는 이유

LLM은 기존의 소규모 모델에서 시도할 수 없던 작업을 수행하는 **새로운 능력**을 보였습니다.

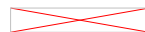
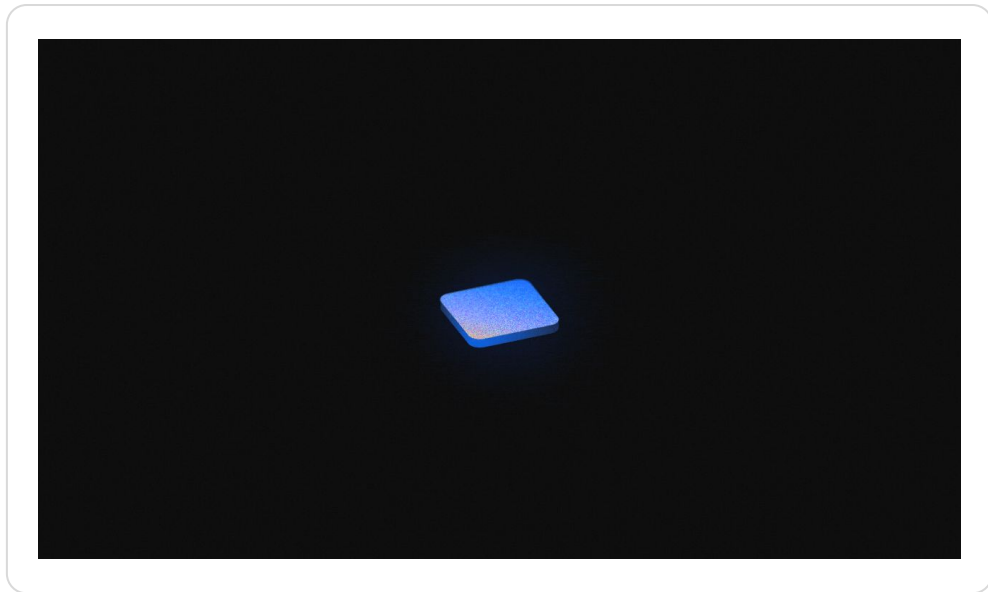
인간 언어에 대한 LLM의 문맥 이해 능력을 통해 우리가 데이터 및 지능형 시스템과 **상호 작용하는 방식이 변화합니다.**

LLM은 **대규모의 이질적인 데이터**에서 패턴과 연관성을 찾을 수 있습니다.

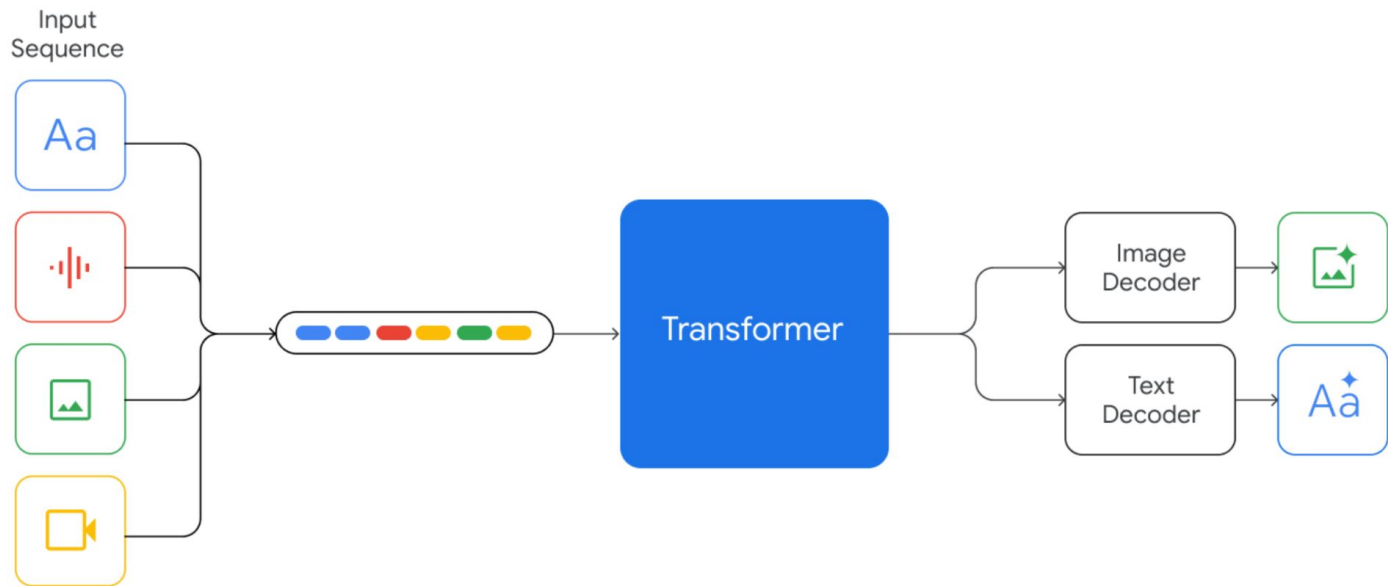


Gemini

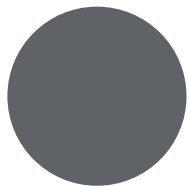
Gemini는 Google이 지금까지 구축한 모델
중 가장 유능한 범용모델로, Google
딥마인드와 Google 리서치를 비롯한 여러
팀이 대규모로 협업한 결과물입니다.



Multimodality



AI 활용안의 발전 방향



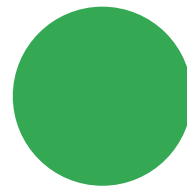
Predictive AI

회귀 및 분류
예측 분석
감성 분석
특성 추출
객체 파악
...



Generative AI

텍스트, 이미지, 코드 생성
텍스트 및 코드 재작성 및 형식화
요약
이미지, 비디오 내용 서술
Q&A 내용 생성
...



Multimodal Generative AI

일반 이미지 이해
공간적 추론 및 논리
시각적 내용에 대한 수학적 추론
영상 질문에 대한 답변
자동 음성 인식 및 통역
...

Build with AI



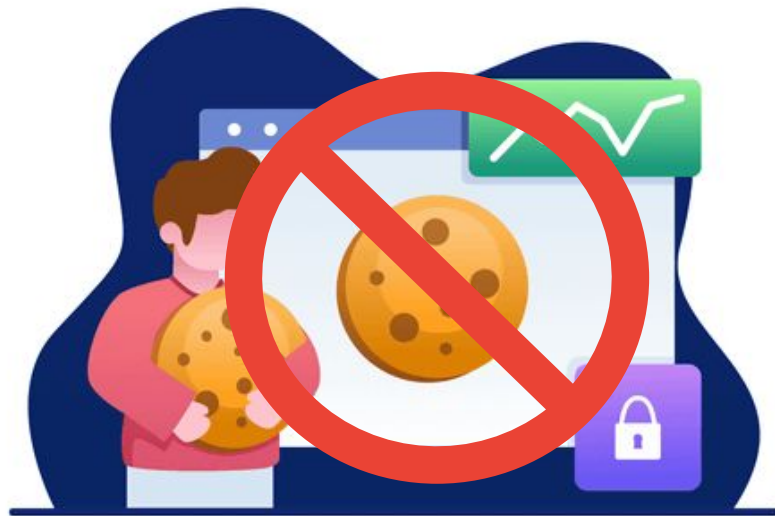
Google Developer Groups
United

Marketing Mix Model

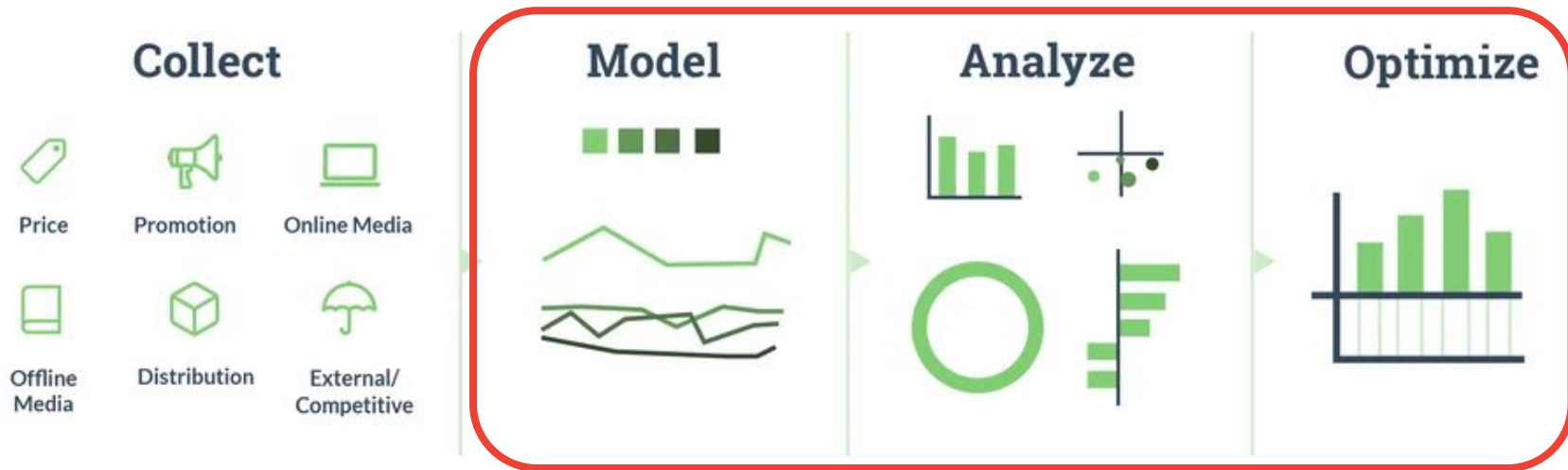
마케팅 환경의 변화

- 브라우저에서 타사 쿠키를 사용하지 못하게 됨
- 보다 강력한 데이터 개인정보 보호 규정이 시행됨

기존의 어트리뷰션 기반 마케팅 성과 측정 및 개별 사용자 수준 데이터 추적이 어려워짐



Marketing Mix Model (MMM)



마케팅 활동을 통해 얼마나 많은 비즈니스 결과(예: 매출)가 발생하는지를 파악하는 방법으로, 투자 대비 수익(ROI) 측면에서 마케팅의 효과를 평가하고 이를 활용하기 위해 만들어진 계량경제학적 접근법

현대적 MMM 주요 접근 방식

Lightweight
MMM



ROBYN



Orbit

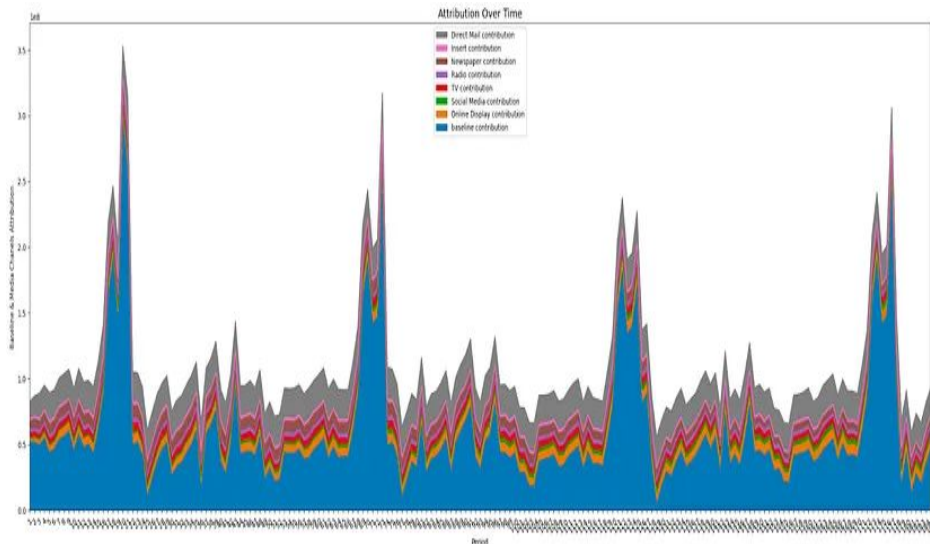
Third Party MMM



만든 곳	Google	Meta	Uber	SaaS 공급업체
성격	오픈소스 MMM 라이브러리 (Python)	오픈소스 MMM 패키지 (R, Python(beta))	오픈소스 시계열 분석 라이브러리 (Python) 활용한 접근	업체별 소프트웨어, 웹서비스 등
활용 특징	<ul style="list-style-type: none"> • 베이지안 확률 모델링 활용 • Numpyro와 JAX를 활용한 빠른 모델링 	<ul style="list-style-type: none"> • 동적 모델링 • 실시간 캠페인 최적화 • 크로스 채널 통합 기능 등 • Prophet 활용 	<ul style="list-style-type: none"> • 베이지안 확률 모델링 활용 • Stan과 Pyro 등을 통합해서 사용 	<ul style="list-style-type: none"> • 일정한 결과 안정성 보장 • 소프트웨어 유지 보수 등

LightWeightMMM

- MMM에 베이지안 접근 방식을 사용하여 광고주가 사전 정보를 모델링에 통합
- 광고주가 데이터를 적절하게 확장하고, 모델을 평가하고, 예산 할당을 최적화하고, 현장에서 사용되는 일반적인 그래프를 그릴 수 있는 기능 제공
- JAX와 NumPyro(JAX 기반 확률 프로그래밍 라이브러리) 기반으로 베이지안 연산을 빠르게 구현





Google Meridian

Empower your team with best-in-class marketing mix models and drive better business outcome

Meridian is an open-source MMM built by Google that provides innovative solutions to key measurement challenges

[Apply for early access](#)

Meridian is currently offered in limited availability



Built for purpose

Meridian was built to enable privacy-durable, advanced measurement while meeting marketers where they are

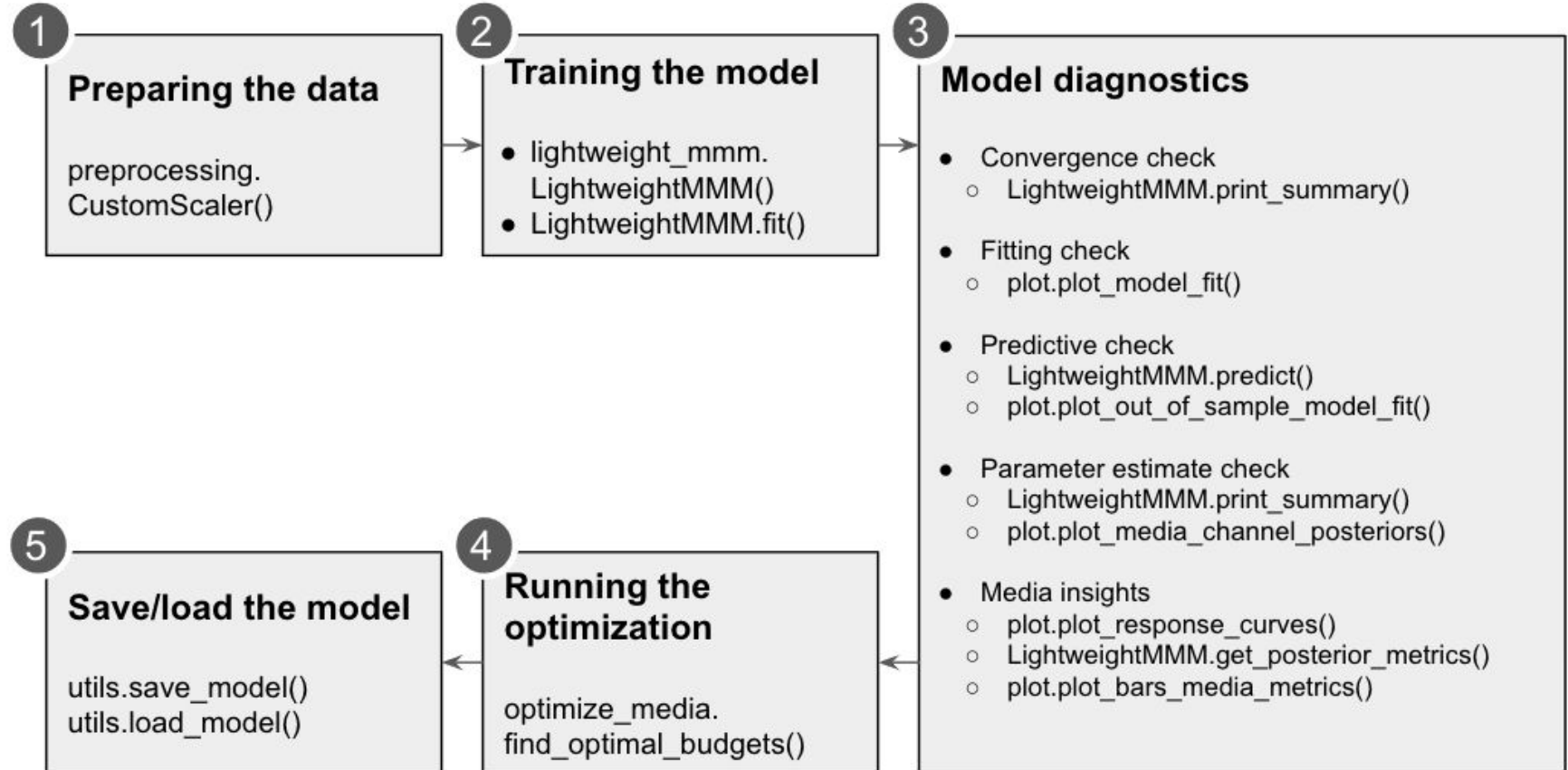


Google Developer Groups

현대적 MMM의 장단점

장점	단점
<ul style="list-style-type: none">● 다양한 마케팅 채널과 접점을 전체적으로 고려하여 통합적인 분석을 제공● 마케팅 활동의 영향을 다차원적으로 머신러닝 등의 기술을 활용하여 분석하고, 예측 등 정교한 분석 가능● 정확하고 광범위한 ROI 계산 가능● 개인정보 및 데이터 관련 외부 요인에서 안정적	<ul style="list-style-type: none">● 활용을 위해 데이터 분석 및 기술적 이해가 필요(높은 학습 곡선)● 데이터가 부족하거나 품질이 낮은 경우 분석 결과의 신뢰성이 떨어질 수 있음

Process of LightweightMMM



Build with AI



Google Developer Groups
United



Marketing Mix Model + Gemini

MMM 예제

- 사용 데이터: Kaggle에 공개된 실제 광고 집행 데이터
(<https://www.kaggle.com/datasets/saicharansirangi/adanalyse>)
 - 월마트 등의 마켓플레이스의 브랜드에 설정된 다양한 광고 그룹 정보)

Campaign Data (9586 rows)

Detail Compact Column 10 of 15 columns

Date	Brand Alias	Ad group alias	ASIN/SKU Alias	Marketplace	# Im
 16Oct21 10Jan22	Brand 2 54%	Brand 2 Ad Group 2 23%	ASIN402 2%	Amazon 33%	 0
	Brand 1 46%	Brand 2 Ad Group 1 21%	ASIN399 2%	Walmart 33%	
		Other (5436) 57%	Other (9202) 96%	Other (3195) 33%	
10/17/2021 00:00:00	Brand 1	Brand 1 Ad Group 10	ASIN414	Walmart	33
10/17/2021 00:00:00	Brand 1	Brand 1 Ad Group 11	ASIN385	Walmart	0
10/17/2021 00:00:00	Brand 1	Brand 1 Ad Group 11	ASIN389	Walmart	0
10/17/2021 00:00:00	Brand 1	Brand 1 Ad Group 13	ASIN377	Walmart	380
10/17/2021 00:00:00	Brand 1	Brand 1 Ad Group 13	ASIN399	Walmart	3805

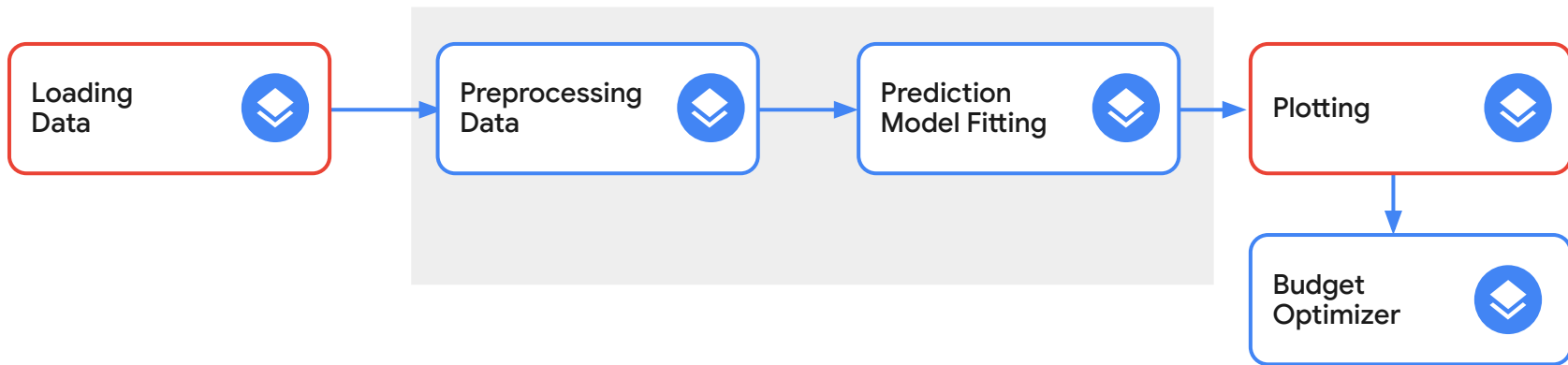
MMM + Gemini 예제

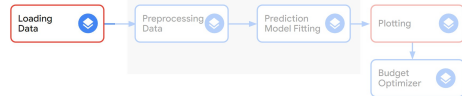
- 사용 환경: Jupyter, BigQuery, Gemini 1.5 Pro
 - BigQuery key 필요
 - Gemini API 키 필요
 - 실행 지역에 따라 실행이 안 되는 경우 있음 (API 사용 가능 지역 확인)
- 사용 라이브러리
 - lightweight-mmm, google-generativeai
 - jax, numpyro, sklearn, pandas, matplotlib
 - Google auth libraries
- 코드 : <https://github.com/cojette/MMMwithGemini/blob/main/AlMMM.ipynb>

Build with AI

사용자가 필요한 부분에 AI를
더하자

기본 실행 내역





Loading Data

```
import glob
from google.cloud import bigquery
from google.oauth2 import service_account

credential = service_account.Credentials.from_service_account_file('

# GCP 클라이언트 객체 생성
client = bigquery.Client(credentials = credential,
                          project = credential.project_id)

## 데이터 가져옴
sql = """
SELECT
  *,
  CASE
    WHEN Date LIKE '% 2021' THEN
      PARSE_DATE('%b %d, %Y', REPLACE(Date, ' ', ''))
    WHEN Date LIKE '%/2021' THEN
      PARSE_DATE('%m/%d/%Y', Date)
    ELSE
      NULL END AS Ad_Date
  FROM adanalysis.campaigndata
"""

query_job = client.query(sql)

# 데이터프레임으로 변환
df = query_job.to_dataframe()
df.head()
```

GCP 인증 후 SQL을 사용해서
BigQuery에서 데이터를 가지고 옴

Raw data

일별 브랜드와 광고 내역 관련된 데이터가 기록되어 있음

	Date	Brand_Alias	Ad_group_alias	ASIN_SKU_Alias	Marketplace	Impressions	Clicks	Spend	Sales	Orders	Units	Advertised_Units_sold
0	Nov 08, 2021	Brand 1	Brand 1 Ad Group 1	ASIN374	Walmart	1168	23	7.37	0.0	0	0	0
1	Nov 12, 2021	Brand 1	Brand 1 Ad Group 1	ASIN374	Walmart	0	0	0.00	0.0	0	0	0
2	Nov 13, 2021	Brand 1	Brand 1 Ad Group 13	ASIN374	Walmart	1712	12	2.18	0.0	0	0	0
3	Nov 14, 2021	Brand 1	Brand 1 Ad Group 13	ASIN374	Walmart	1014	4	0.70	0.0	0	0	0
4	Nov 19, 2021	Brand 1	Brand 1 Ad Group 13	ASIN374	Walmart	1308	17	3.11	0.0	0	0	0

Preprocessing Data



```
## Data Agregation
agg_data = df.groupby(["Ad Date", "Ad_group_alias"])[["Impressions", "Spend", "Sales",]].sum()
agg_data = agg_data.drop(["Brand 1 Ad Group 12"], axis=0, level=1) # zero cost train

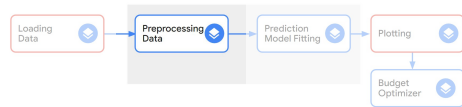
media_data_raw = agg_data['Impressions'].unstack().fillna(0)
costs_raw = agg_data['Spend'].unstack()
sales_raw = agg_data['Sales'].reset_index().groupby("Ad Date").sum()
```

데이터를 날짜와 AdGroup 기준으로
집계하고 이상치 처리

```
import numpy as np

## Make random noises
organic_raw = pd.DataFrame({'organic_search': 0, 'organic_social': 0}, index=media_data_raw.index)
organic_raw['organic_search'] = sales_raw['Sales'].values/10 + np.random.randint(10000, 100000, organic_raw.shape[0])
organic_raw['organic_social'] = sales_raw['Sales'].values/10 + np.random.randint(10000, 100000, organic_raw.shape[0])
```

Scaling Data



```
## Divide Training data and test data with timestamp
split_point = pd.Timestamp("2021-12-15") # 28 days to end of data

media_data_train = media_data_raw.loc[:split_point - pd.Timedelta(1,'D')]
media_data_test = media_data_raw.loc[split_point:]

organic_data_train = organic_raw.loc[:split_point - pd.Timedelta(1,'D')]
organic_data_test = organic_raw.loc[split_point:]

target_train = sales_raw.loc[:split_point - pd.Timedelta(1,'D')]
target_test = sales_raw.loc[split_point:]

costs_train = costs_raw.loc[:split_point - pd.Timedelta(1,'D')].sum(axis=0).loc[media_data_train.columns]

## Scale data with mean()
media_scaler = preprocessing.CustomScaler(divide_operation=jnp.mean)
organic_scaler = preprocessing.CustomScaler(divide_operation=jnp.mean)
target_scaler = preprocessing.CustomScaler(
    divide_operation=jnp.mean)
cost_scaler = preprocessing.CustomScaler(divide_operation=jnp.mean)

### data transformation for model
media_data_train_scaled = media_scaler.fit_transform(media_data_train.values.astype(int))
organic_data_train_scaled = organic_scaler.fit_transform(organic_data_train.values)
target_train_scaled = target_scaler.fit_transform(target_train['Sales'].values.squeeze())
costs_scaled = cost_scaler.fit_transform(costs_train.values.astype(int))

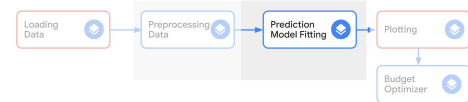
media_data_test_scaled = media_scaler.transform(media_data_test.values.astype(int))
organic_data_test_scaled = organic_scaler.transform(organic_data_test.values)

media_names = media_data_raw.columns
```

특정 시간을 기준으로 이전까지의
광고 데이터를 훈련 데이터로, 이후
데이터를 검정 데이터로 구분

평균을 기준으로 데이터를 축소함

모델에 대입할 숫자값으로 변환



Prediction Model Fitting

```
#adstock_models = ["adstock", "hill_adstock", "carryover"]
#degrees_season = [1,2,4]

adstock_models = ["carryover"]
degrees_season = [1]

## model fitting
for model_name in adstock_models:
    for degrees in degrees_season:
        mmm = lightweight_mmm.LightweightMMM(model_name=model_name)
        mmm.fit(media=media_data_train_scaled,
                media_prior=costs_scaled,
                target=target_train_scaled,
                extra_features=organic_data_train_scaled,
                number_warmup=10,
                number_samples=1000,
                number_chains=1,
                degrees_seasonality=degrees,
                weekday_seasonality=True,
                seasonality_frequency=365,
                seed=1)

## Prediction
prediction = mmm.predict(
    media=media_data_test_scaled,
    extra_features=organic_data_test_scaled,
    target_scaler=target_scaler)
p = prediction.mean(axis=0)

## Calculate error rate
mape = mean_absolute_percentage_error(target_test['Sales'].values, p)
print(f"model_name={model_name} degrees={degrees} MAPE={mape} samples={p[:3]}")
```

인과 관계 합성곱을 적용하여 먼 값보다 가까운 값에 더 많은 가중치를 부여하는 모델

Model fitting

Prediction

```

prediction = mmm.predict(
    media=media_data_test_scaled,
    extra_features=organic_data_test_scaled,
    target_scaler=target_scaler)
p = prediction.mean(axis=0)

mape = mean_absolute_percentage_error(target_test['Sales'].values, p)
print(f"model_name={model_name} degrees={degrees} MAPE={mape} samples={p[:3]}")

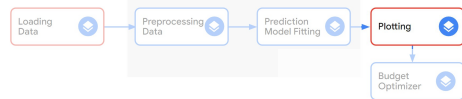
```

```

sample: 100%|██████████| 1010/1010 [01:20<00:00, 12.60it/s, 1023 steps of size 2.52e-03, acc. prob=0.89]
model_name=adstock degrees=1 MAPE=0.12187085696205786 samples=[29713.23 30937.893 29248.553]
sample: 100%|██████████| 1010/1010 [01:20<00:00, 12.60it/s, 1023 steps of size 2.52e-03, acc. prob=0.89]
model_name=adstock degrees=2 MAPE=0.18745539910990147 samples=[28746.727 29699.748 27948.379]
sample: 100%|██████████| 1010/1010 [01:13<00:00, 13.76it/s, 1023 steps of size 2.53e-03, acc. prob=0.81]
model_name=adstock degrees=4 MAPE=0.42127081538802974 samples=[27145.957 26536.264 24296.111]
sample: 100%|██████████| 1010/1010 [01:11<00:00, 14.05it/s, 831 steps of size 2.45e-03, acc. prob=0.89]
model_name=hill_adstock degrees=1 MAPE=0.15217211133335748 samples=[32458.893 32559.52 33259.703]
sample: 100%|██████████| 1010/1010 [01:11<00:00, 14.04it/s, 1023 steps of size 2.34e-03, acc. prob=0.90]
model_name=hill_adstock degrees=2 MAPE=0.14894072706727735 samples=[31663.607 31239.338 31270.371]
sample: 100%|██████████| 1010/1010 [01:11<00:00, 14.22it/s, 18 steps of size 2.46e-03, acc. prob=0.88]
model_name=hill_adstock degrees=4 MAPE=0.2129068249582706 samples=[32078.34 32597.63 32926.906]
sample: 100%|██████████| 1010/1010 [18:02<00:00, 1.07s/it, 1023 steps of size 2.53e-03, acc. prob=0.93]
model_name=carryover degrees=1 MAPE=0.09326828947397732 samples=[30079.762 29410.35 29076.996]
sample: 100%|██████████| 1010/1010 [17:59<00:00, 1.07s/it, 1023 steps of size 2.52e-03, acc. prob=0.92]
model_name=carryover degrees=2 MAPE=0.3047367384459012 samples=[27912.648 26419.623 24849.475]
sample: 100%|██████████| 1010/1010 [17:44<00:00, 1.05s/it, 1023 steps of size 2.53e-03, acc. prob=0.91]
model_name=carryover degrees=4 MAPE=0.3403701302182601 samples=[27523.098 25486.15 23824.072]

```

3개의 모델에 대해 각각 3개의 degree를 넣어 확인 결과 MAPE(평균 절대 비율 오차)가 확연하게 적은 모델과 degree 선택



Plotting

```
## plot the prior and posterior distributions for every model parameter (It's very long and I chose only one feature)
!pip install matplotlib==3.1.3

plt = plot.plot_prior_and_posterior(media_mix_model=mmm, number_of_samples_for_prior = 1000, selected_features = ['ad_effect_retention_rate'])
fig = plt.get_figure()

...

fig.savefig("plot_media_posterior.png", bbox_inches = "tight")

# plot the estimated media contributions with their respective credibility intervals

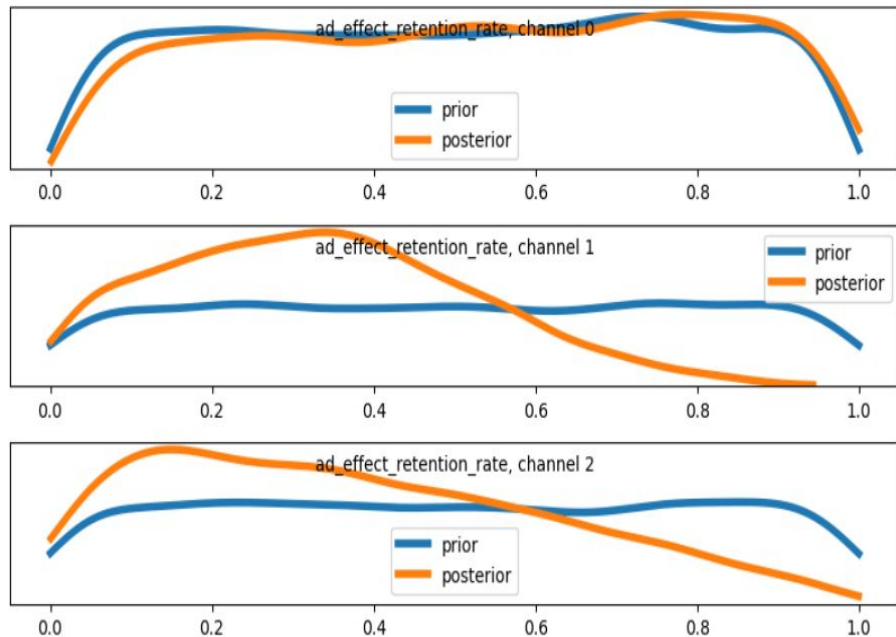
media_effect_hat, roi_hat = mmm.get_posterior_metrics()
plt = plot.plot_bars_media_metrics(metric = media_effect_hat, channel_names=media_names)
fig = plt.get_figure()
fig.savefig("plot_media_hat.png", bbox_inches = "tight")
plt = plot.plot_bars_media_metrics(metric = roi_hat, channel_names=media_names)
fig = plt.get_figure()
fig.savefig("plot_roi_hat.png", bbox_inches = "tight")

...

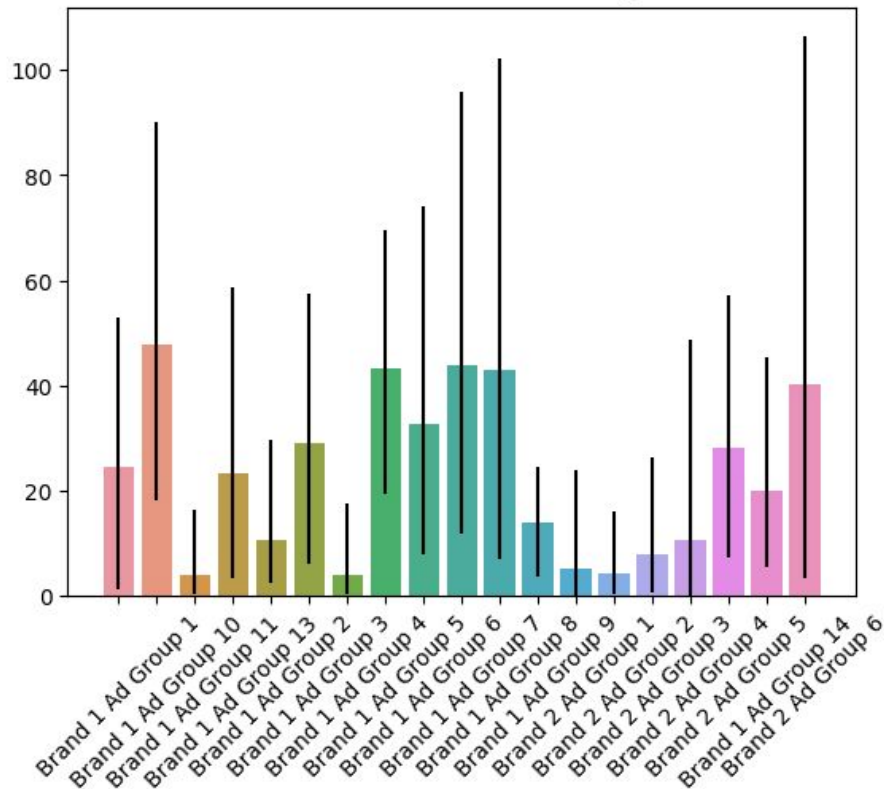
## plot the curve response of all media channels that how each media channel behaves individually as we invest more in it.
plt = plot.plot_response_curves(
    media_mix_model=mmm, target_scaler=target_scaler, seed=1)
fig = plt.get_figure()
fig.savefig("plot_res_curve.png", bbox_inches = "tight")
```

LightWeightMMM에서 자체 그래프
기능 지원
(pandas 기반 선/막대 그래프)

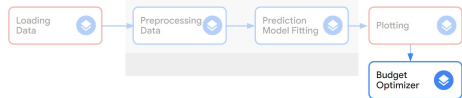
Plotting



Estimated media channel metric.
Error bars show 0.05 - 0.95 credibility interval.



Budget Optimizer



```
# Run optimization with the parameters of choice.
solution, kpi_without_optim, previous_media_allocation = optimize_media.find_optimal_budgets(
    n_time_periods=n_time_periods,
    media_mix_model=mmm,
    # extra_features=extra_features_scaler.transform(extra_features_test)[:n_time_periods],
    budget= sum(budget.values),
    prices=prices,
    media_scaler=media_scaler,
    target_scaler=target_scaler,
    max_iterations = 10,
    seed=1)
```

```
Iteration limit reached      (Exit mode 9)
      Current function value: -1251056.5948636667
      Iterations: 10
      Function evaluations: 390
      Gradient evaluations: 10
```

```
optimal_buget_allocation = prices * solution.x
optimal_buget_allocation
```

```
Array([1.8141009e+05, 1.5976271e+03, 3.5399262e+06, 9.6178794e+05,
       1.2558705e+06, 1.6166907e+04, 2.2025928e+07, 2.3277966e+03,
       2.2146145e+04, 3.0611891e+04, 2.5923617e+04, 4.4037706e+05,
       1.7143188e+07, 2.7503890e+07, 5.2502645e+06, 1.2654550e+06,
       9.4129088e+05, 4.5191903e+05, 7.2498643e+03], dtype=float32)
```

```
# similar renormalization to get previous budget allocation
previous_budget_allocation = prices * previous_media_allocation
previous_budget_allocation
```

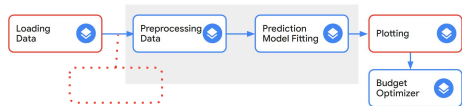
```
Array([1.8144233e+05, 1.3068665e+03, 3.5400885e+06, 9.6188712e+05,
       1.2560151e+06, 1.5516023e+04, 2.2026096e+07, 1.9041487e+03,
       2.1957184e+04, 3.0444746e+04, 2.5856990e+04, 4.4047891e+05,
       1.7143354e+07, 2.7504056e+07, 5.2504270e+06, 1.2656099e+06,
       9.4143306e+05, 4.5207431e+05, 7.3828384e+03], dtype=float32)
```

3개의 모델에 대해 각각 3개의 degree를 넣어 확인 결과 MAPE(평균 절대 비율 오차)가 확연하게 적은 모델과 degree 선택

Build with AI

사용자가 필요한 부분에 AI를
더하자

Understanding Data with Gemini



```
genaimodel = genai.GenerativeModel(model_name="gemini-1.5-pro-latest")

# 프롬프트 설명 : 데이터에 대한 간단한 배경, 앞으로 활용 용도(MMM), EDA 및 outlier에 대한 의견 요청
prompt_parts = "This data has information about different ad groups that were set for brands in a few marketplaces like Walmart."

# 모델 설정
generation_config = {
    "temperature": 2,
    "top_p": 0.95,
    "top_k": 0,
    "max_output_tokens": 9999
}

# 데이터와 함께 해당 프롬프트를 Gemini에 전달
response = genaimodel.generate_content([prompt_parts, df.to_csv()], generation_config=generation_config)
```

데이터에 대한 간단한 배경과 활용
용도 설명 후 EDA 및 outlier 정보
요청

```
## 한글 번역 요청
```

```
response2 = genaimodel.generate_content(["다음 내용을 한국어로 바꿔줘. ", response.text], generation_config=generation_config)
```

```
import textwrap
```

```
from IPython.display import display
from IPython.display import Markdown
```

```
def to_markdown(text):
    text = text.replace('•', ' *')
```

광고 그룹 실적에 대한 탐색적 데이터 분석 (EDA)

제공된 데이터를 기반으로, 잠재적 이상치에 중점을 둔 초기 EDA는 다음과 같습니다.

요약 통계:

- **노출 수:** 광고 그룹당 평균 노출 수는 약 9,943회이고, 중앙값은 2,889회입니다. 이는 몇몇 높은 노출 수를 가진 광고 그룹의 영향으로 인해 분포가 왜곡되었을 가능성을 나타냅니다.
- **클릭 수:** 평균 클릭 수는 26회이지만, 중앙값은 9회에 불과하여 다시 한 번 오른쪽으로 치우친 분포를 암시합니다. 대부분의 광고 그룹은 클릭 수가 적고, 일부만이 많은 클릭 수를 유도하는 것으로 보입니다.
- **지출:** 평균 지출은 33.64이고, 중앙값은 12.82로, 역시 오른쪽으로 치우친 분포를 보입니다. 이는 대부분의 광고 그룹은 상대적으로 지출이 적지만, 일부는 상당히 높은 지출을 하는 것으로 해석할 수 있습니다.
- **판매 및 주문:** 이 두 지표는 평균값이 중앙값보다 훨씬 높아, 분포가 매우 치우쳐 있음을 나타냅니다. 몇몇 광고 그룹이 상당한 판매 및 주문을 유도하는 반면, 더 큰 비중은 최소한의 판매/주문을 생성하거나 아예 생성하지 않습니다.
- **판매된 단위 및 광고된 판매 단위:** 판매 및 주문과 마찬가지로, 판매된 단위와 광고된 판매 단위 모두 평균값이 중앙값보다 상당히 큼니다. 이는 판매 및 주문과 유사한 오른쪽으로 치우친 패턴을 암시합니다. 일부 광고 그룹은 많은 판매를 하지만, 많은 광고 그룹은 판매량이 매우 적거나 전혀 없습니다.
- **기타 지표:** 기타 SKU 판매 단위, 광고된 SKU 판매 및 기타 SKU 판매는 판매 및 주문과 유사한 패턴을 보이는 것 같습니다. 이는 값이 매우 높은 이상치 행이 존재할 가능성을 나타냅니다.

잠재적 이상치:

잠재적 이상치를 식별하기 위해 여러 가지 접근 방식을 고려할 수 있습니다.

1. **노출/클릭/지출이 0인 행:** 노출, 클릭 및 지출이 0이지만 판매 및 주문은 0이 아닌 행이 많이 있습니다. 이러한 행은 데이터 기록 오류 또는 직접적인 광고 참여 없이 발생한 판매(유기적 판매)를 나타낼 수 있습니다. 이를 추가로 조사하는 것이 좋습니다.
2. **왜곡이 심한 지표:** 판매, 주문, 판매 단위와 같은 지표의 경우, 값이 75번째 백분위수보다 상당히 높거나 사분위수 범위의 3배를 초과하는 행을 잠재적 이상치로 간주해야 합니다. 이러한 성과가 높은 광고 그룹의 맥락을 파악하는 것이 중요합니다.

추가 EDA 단계:

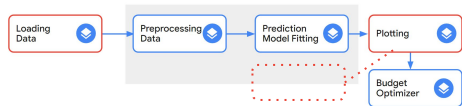
1. **시각화:** 노출 수, 클릭 수, 지출, 판매, 주문과 같은 주요 지표에 대한 히스토그램 또는 상자 그림을 생성하여 분포를 더 잘 이해하고 이상치 행을 시각적으로 식별합니다.
2. **세분화:** Marketplace, Brand_Alias, Ad_group_alias별로 실적을 분석하여 잠재적인 차이점을 파악하고 각 세그먼트 내에서 이상치를 식별합니다.
3. **상관 관계:** 주요 지표 간의 상관 관계를 조사합니다. 예를 들어, 지출이 많을수록 판매/주문이 증가하는지 또는 특정 지출 임계값을 초과하면 수익이 감소하는지 확인합니다.

이상치 활용:

- **맥락 조사:** 일부 광고 그룹에서 값이 매우 높거나 낮은 이유를 이해합니다. 특정 캠페인 설정, 타겟 잠재 고객, 계절성, 제품 특성 또는 외부 요인 때문일 수 있습니다.
- **잠재적 제거:** 데이터 오류로 인해 이상치가 발생했거나 마케팅 믹스 모델과 관련이 없는 것으로 간주되는 경우, 일반적인 광고 그룹 실적을 더 정확하게 나타내기 위해 이상치를 제거하거나 조정하는 것을 고려합니다.
- **모델링 고려 사항:** 마케팅 믹스 모델에서는 이상치에 강건한 분위 회귀 또는 강건 회귀와 같은 다양한 모델링 기술을 탐구하는 것이 좋을 수 있습니다.

포괄적인 EDA를 수행하고 잠재적 이상치 행을 신중하게 분석하면 광고 그룹 실적에 대한 귀중한 통찰력을 얻고 마케팅 믹스 모델이 안정적인 기반 위에 구축되도록 할 수 있습니다.

Plotting Description with Gemini



```
from pathlib import Path
import hashlib

uploaded_files = []
def upload_if_needed(pathname: str) -> list[str]:
    path = Path(pathname)
    hash_id = hashlib.sha256(path.read_bytes()).hexdigest()
    try:
        existing_file = genai.get_file(name=hash_id)
        return [existing_file]
    except:
        pass
    uploaded_files.append(genai.upload_file(path=path, display_name=hash_id))
    return [uploaded_files[-1]]
```

```
## 프롬프트: 앞에서 그린 그래프에 대해 간략히 설명하고 이를 그래프 4종류를 업로드함. 이를 바탕으로 결과를 해석하는 것을 요청함
```

```
prompt_parts = [
    "These graphs are from marketing mix model result.",
    "First graph shows an visualize the prior and posterior distributions for every channel that ads effect to the retention rate at once. Note that we use a ke",
    "Second plot is also about media effect metrics. Third plot is the ROI plot. These plots take into account not only the media effect but how much it costs t",
    "For that we plot the curve response of all media channels. It is the fourth plot. ",
    " I hope you read and understand these graph and write a report to help us make decisions about our marketing advertising budget.",
    *upload_if_needed('plot_media_posterior.png'),
    *upload_if_needed("plot_media_hat.png"),
    *upload_if_needed("plot_roi_hat.png"),
    *upload_if_needed("plot_res_curve.png"),
    "Please be as helpful as possible in describing your interpretation of the graph and what you can do with it. Thanks in advance. ",
]
response = genaimodel.generate_content(prompt_parts, generation_config=generation_config)
to_markdown(response.text)
for uploaded_file in uploaded_files:
    genai.delete_file(name=uploaded_file.name)
```

MMM에서 만든 결과 그래프에 대한 배경 지식을 제공한 후 해당 그래프를 읽고 이에 대해서 설명하고 인사이트를 찾아내는 것을 요청

```
response2 = genaimodel.generate_content(["다음 내용을 한국어로 바꿔줘. ", response.text], generation_config=generation_config)
to_markdown(response2.text)
```


마케팅 믹스 모델 그래프를 통한 인사이트

제공된 그래프를 기반으로 마케팅 광고 예산에 대한 인사이트와 잠재적 행동 방안을 제시합니다.

1. 리텐션율에 대한 채널 효과

- **상당한 변화:** 첫 번째 그래프 세트는 여러 채널(예: 0, 1, 2, 3, 4, 7, 8, 10, 12)에 대해 사전 및 사후 분포에서 상당한 변화를 나타냅니다. 이는 이러한 채널이 사용자 리텐션에 상당한 영향을 미친다는 것을 시사합니다.
- **영향력이 큰 채널에 집중:** 사후 분포 변화가 큰 채널에 예산을 더 많이 할당합니다. 이러한 채널은 사용자 리텐션 측면에서 투자 대비 가장 효과적인 수익을 제공할 가능성이 높습니다.
- **영향력이 낮은 채널 분석:** 변화가 적은 채널(예: 5, 6, 9, 11, 13-18)의 실적이 저조한 이유를 조사합니다. 잘못된 타겟층을 타겟팅하고 있는지, 크리에이티브 개선이 필요한지 고려합니다. 예산 할당을 줄이거나 제거하는 것도 고려할 수 있습니다.

2. 미디어 채널 메트릭

- **다양한 성과:** 두 번째 및 세 번째 그래프는 다양한 브랜드 및 광고 그룹에 걸쳐 다양한 성과를 보여줍니다.
- **최고 실적 채널 식별:** 브랜드 1 광고 그룹 2, 8, 9, 브랜드 2 광고 그룹 1, 브랜드 1 광고 그룹 13이 특히 효과적인 것으로 보입니다. 이러한 최고 실적 그룹에 대한 투자를 늘려 수익을 극대화합니다.
- **저성과 채널 조사:** 브랜드 2 내의 여러 그룹(그룹 1 제외)이 저조한 성과를 보이는 것으로 보입니다. 타겟팅, 크리에이티브 또는 플랫폼 선택과 관련된 잠재적 문제를 분석합니다.

3. 투자 수익률(ROI)

- **비용 효율성:** 미디어 효과 메트릭을 관련 비용과 비교하여 ROI를 확인합니다.
- **ROI가 높은 채널 우선 순위 지정:** 비용 대비 더 높은 영향을 미치는 채널에 더 많은 예산을 할당합니다.
- **ROI가 낮은 채널 평가:** 영향력이 적은 채널의 가치를 재평가합니다. 개선하거나 더 효율적인 대안으로 교체하는 것을 고려합니다.

4. 반응 곡선 및 포화점

- **수익 감소:** 반응 곡선은 대부분의 채널에서 지출이 증가함에 따라 수익이 감소한다는 것을 나타냅니다. 곡선이 평평해지기 시작하는 지점을 식별합니다. 이는 추가 지출이 최소한의 추가 가치만 제공하는 포화점을 나타냅니다.
- **예산 할당 최적화:** 반응 곡선을 기반으로 채널 전체에 예산을 효과적으로 분배하여 포화된 채널에 과도하게 지출하지 않도록 합니다. 포화점에 도달하기 전에 성장 가능성이 더 큰 채널에 리소스를 할당하는 것을 고려합니다.

권장 사항:

- **영향력이 큰 채널 우선 순위 지정:** 리텐션에 명확한 영향을 미치고 높은 ROI를 제공하는 채널에 집중합니다.
- **저성과 채널 분석 및 개선:** 저성과 이유를 조사하고 타겟팅, 크리에이티브 또는 플랫폼 선택을 조정하는 것을 고려합니다. 개선이 불가능한 경우 예산을 줄이거나 재할당하는 것을 고려합니다.

Build with AI

사람들을 널리 이롭게 하려면
사람들의 필요를 먼저 파악하라

Further Action Items

- Meridian 접근 가능할 때까지 기다린다
- AI 연계 추가
 - 더 쉬운 모델 선택
 - 매개변수 조정 등을 사용자가 직접 할 수 있는 방안 고안
 - 보다 상세한 데이터 정제 방안 제시 방법 등
- 타 라이브러리 기능에 AI 연계
- 타 분야의 더 쉬운 데이터 활용을 위한 AI 연계 고민

- 각자의 분야 개선
 - 초심자나 각자의 분야에서 잘 모르는 사람들을 위한 AI 활용 방안 고안
 - 작은 것부터 이야기하고 시작해보기

Build with AI

좋은 것을 잘 더해서
탁월함을 만들자

Build with AI



Google Developer Groups
United

Thank you!