Marketing Mix Modeling with Meta Robyn

Sheryl Xu, Master of Science @ Northwestern University

2025-07-27

Part I: Introduction

Meta Robyn Meta's Robyn is an open-source, automated Marketing Mix Modeling (MMM) tool that helps marketers quantify the effectiveness of various media channels and make data-driven budget allocation decisions. It combines advanced statistical modeling with Bayesian optimization to evaluate channel contributions, model saturation and adstock effects, and generate multiple candidate models along a Pareto frontier.

Robyn is particularly valuable because it brings transparency, scalability, and automation to a traditionally manual and resource-intensive process—enabling more agile marketing strategies in a privacy-centric world where granular user-level tracking becomes increasingly limited.

This project presents a time-series Marketing Mix Modeling (MMM) analysis using Meta's Robyn package, with the objective of quantifying the impact of various marketing channels and informing optimized media budget allocation.

The analysis uses a synthetic dataset originally developed by **Professor Elea Feit** of Drexel University for educational purposes. Three CSV files form the foundation of this project:

- customer.csv, which contains user profile information.
- impressions.csv, detailing ad exposures and clicks by date and channel.
- transactions.csv, which records the daily transactions made by users.

Data Import and Overview

```
customer <- fread("/Users/itsxuuu/Desktop/MMM/Data/customer.csv")
impressions <- fread("/Users/itsxuuu/Desktop/MMM/Data/impressions.csv")
transactions <- fread("/Users/itsxuuu/Desktop/MMM/Data/transactions.csv")
```

```
head(customer)
```

```
##       id past.purchase email direct
##    <int>         <int> <int>  <int>
## 1:     1             0     0      1
## 2:     2             1     1      1
## 3:     3             1     1      0
## 4:     4             0     0      0
## 5:     5             1     1      1
## 6:     6             0     0      0
```

- `customer.csv`: Information about the users with some fields from their profiles. Each row in the file represents a customer, 10,000 rows; the columns describe some basic information about each customer including id number, whether the customer has made a purchase prior to the observation period, and whether the customer is eligible to receive emails or direct mails.

```
head(impressions)
```

```
##       id     date channel click
##    <int>   <char>  <char> <int>
## 1:     1 1/6/2017  direct     0
## 2:     1 2/3/2017  direct     0
## 3:     1 1/1/2017  social     0
## 4:     1 1/2/2017  social     0
## 5:     1 1/5/2017  social     0
## 6:     1 1/6/2017  social     0
```

- `impressions.csv`: Information about which users are connected to other users. Each row is an exposure of marketing communication to a specific customer, 501,336 rows; the columns describe the information about the customer's impression towards an advertisement including id number for the customer, date of impression, the channel of the ad exposure, and whether the customer clicked on the ad.

```
head(transactions)
```

```
##       V1    id      date last.touch last.click
##    <int> <int>    <char>     <char>     <char>
## 1:     1     2  1/4/2017      email       none
## 2:     2     2 2/12/2017      email       none
## 3:     3     3  2/2/2017      email       none
## 4:     4     3 2/14/2017      email       none
## 5:     5     5  1/4/2017    display      email
## 6:     6     5 1/13/2017    display      email
```

- `transactions.csv`: Information about history of the user's account registrations (logins) over time. Each row is a transaction made by a customer; columns record the basic information of a transaction including customer id, date of the transaction, channel of the last ad impression the customer saw before the transaction, and channel of the last ad the customer clicked before the transaction.

Part II: Data Aggregation and Preparation

Aggregation is a critical step in preparing the data for Marketing Mix Modeling (MMM) using Meta's Robyn package. This process is necessary for the following reasons:

- **Time-series requirement:** Robyn expects input data to be in a time-series format (daily, weekly, or monthly). Aggregating the data ensures each row represents a consistent time unit—typically one day in this case.
- **Aligning sales with media activity:** Aggregation allows link daily sales outcomes with corresponding daily impressions and clicks across different marketing channels, creating a unified dataset suitable for modeling.

- **Capturing zero-activity days:** By joining aggregated impressions and transactions, the merged dataset retains days with media exposure but no conversions (or vice versa), which is essential for maintaining data continuity and ensuring a complete view of marketing activity over time.

```
daily_impressions <- impressions %>%
  group_by(date, channel) %>%
  summarise(
    impressions = n(),
    clicks = sum(click, na.rm = TRUE),
    .groups = "drop") %>%
  pivot_wider(
    names_from = channel,
    values_from = c(impressions, clicks),
    values_fill = 0)

daily_sales <- transactions %>%
  group_by(date) %>%
  summarise(sales = n(), .groups = "drop")

daily_data <- full_join(daily_sales, daily_impressions, by = "date") %>%
  arrange(date) %>%
  replace(is.na(.), 0)
daily_data$date <- as.Date(daily_data$date, format = "%m/%d/%Y")
daily_impressions$date <- as.Date(daily_impressions$date, format = "%m/%d/%Y")
daily_sales$date <- as.Date(daily_sales$date, format = "%m/%d/%Y")
```

Hyperparameter Tuning

This section defines the **hyperparameter search space for each media channel**, allowing Robyn to explore a wide range of possible adstock and saturation dynamics.

Hyperparameters are essential in Robyn because they guide the model in capturing **how advertising effects decay over time (adstock)** and **how returns diminish as impressions increase (saturation)**. Without appropriate hyperparameter tuning, the model might underfit or overfit, leading to misleading conclusions.

By setting realistic bounds for each channel's parameters, the optimization process becomes both statistically sound and reflective of expected media behavior—ultimately improving the accuracy and interpretability of the final model.

```
hyperparameters <- list(
  impressions_direct_alphas = c(0.5, 3),
  impressions_direct_gammas = c(0.3, 1.0),
  impressions_direct_thetas = c(0, 0.3),

  impressions_display_alphas = c(0.5, 3),
  impressions_display_gammas = c(0.3, 1.0),
  impressions_display_thetas = c(0, 0.3),

  impressions_email_alphas = c(0.5, 3),
  impressions_email_gammas = c(0.3, 1.0),
  impressions_email_thetas = c(0, 0.3),

  impressions_social_alphas = c(0.5, 3),
```

```
  impressions_social_gammas = c(0.3, 1.0),
  impressions_social_thetas = c(0, 0.3)
)
```

For each media channel (impressions_direct, impressions_display, impressions_email, impressions_social), it specifies ranges for three key hyperparameters:

- **alpha:** controls saturation, which is how quickly media impact diminishes as spend increases. Higher values allow for slower saturation. `c(0.5, 3)` allows the model to test both low and high saturation.
- **gamma:** shapes the response curve, controlling how steeply response increases with input. `c(0.3, 1.0)` captures the true response dynamics of each channel without forcing overly aggressive or flat curves.
- **theta:** controls the adstock decay, which is how long the impact of media lasts over time. A higher theta = longer carryover effect. `c(0, 0.3)` allows no carryover to up to ~30% of the media effect to carry into the next period, especially for short-duration digital media.

Create InputCollect Object

To initiate the modeling process, the `InputCollect` object is constructed using the `robyn_inputs()` function. This object serves as the foundational configuration for the Robyn model, consolidating the cleaned and aggregated dataset (daily_data), variable mappings (e.g., date, dependent variable, and media channels), modeling timeframe, adstock function, and predefined hyperparameter ranges.

```
InputCollect <- robyn_inputs(
  dt_input = daily_data,
  date_var = "date",
  dep_var = "sales",
  dep_var_type = "conversion",
  paid_media_vars = c("impressions_display", "impressions_social", "impressions_email", "impressions_di
  paid_media_spends = c("impressions_display", "impressions_social", "impressions_email", "impressions_
  window_start = min(daily_data$date),
  window_end = max(daily_data$date),
  adstock = "geometric",
  hyperparameters = hyperparameters
)
```

Model Training with Automated Search

Robyn's `robyn_run()` function is a critical step in the modeling process, as it initiates the automated search for optimal model configurations based on the defined hyperparameter space.

This function leverages **Nevergrad's evolutionary optimization** to generate and evaluate hundreds of model candidates across a multi-dimensional space of adstock rates, saturation curves, and channel effects. Each model is scored using **various diagnostic metrics** (e.g., R-squared, NRMSE, MAPE) to identify those that offer the best balance between fit and generalizability.

Due to GPU limitations, the modeling process is executed with conservative settings: 100 iterations, 200 trials, and a single core. While this limited the total number of models explored, it is sufficient to yield **a diverse set of high-performing candidates**, enabling meaningful decomposition of media contributions and ROI estimation. This automated model selection process ensures objectivity, consistency, and transparency in determining which media strategies are most effective.

```
OutputModels <- robyn_run(
  InputCollect = InputCollect,
  hyperparameters = hyperparameters,
  iterations = 100,
  trials = 200,
  cores = 1
)
```

Part III: Model Evaluation and Interpretation

Extract and Filter Final Model Candidates

The input objects OutputModels and InputCollect contain all the candidate models generated during the `robyn_run()` step and the preprocessed data and parameters. The parameters `pareto_fronts = 3` and `min_candidates = 10` tell Robyn to retain the top-performing models from **up to three Pareto fronts**, ensuring a balance between performance and model complexity, while limiting the output to at least ten candidate models.

```
OutputCollect <- robyn_outputs(
  OutputModels = OutputModels,
  InputCollect = InputCollect,
  export = FALSE,
  pareto_fronts = 3,
  min_candidates = 10
)
```

```
ls(OutputCollect)
```

```
##  [1] "add_penalty_factor"     "allPareto"              "allSolutions"
##  [4] "calibration_constraint" "clusters"               "cores"
##  [7] "hyper_fixed"            "hyper_updated"          "intercept"
## [10] "intercept_sign"         "iterations"             "mediaVecCollect"
## [13] "nevergrad_algo"         "OutputModels"           "pareto_fronts"
## [16] "plot_folder"            "resultCalibration"      "resultHypParam"
## [19] "seed"                   "trials"                 "UI"
## [22] "xDecompAgg"             "xDecompVecCollect"
```

The `OutputCollect` object contains all essential results from the Robyn modeling process, including **top-performing models, channel-level contributions, hyperparameter settings, and diagnostics**. Key components like `xDecompAgg` and `xDecompVecCollect` allow for detailed interpretation of media effectiveness over time, while performance metrics and calibration details support model validation. This structured output enables data-driven decision-making and transparent marketing optimization.

Top Model Summary

During the `robyn_run()` phase, Robyn creates hundreds of candidate models using randomized combinations of hyperparameters (e.g., adstock decay, saturation levels). However, not all models are equally good — some may overfit, underfit, or have unstable decompositions so it's improtant to identify the best-performing Marketing Mix Models (MMMs) from all the candidates Robyn has generated during the training process.

The top model (`solID = 151_96_1`) has the best fit, explaining ~**69.7% of sales variation** (`rsq_train = 0.697`) with **low prediction error** (`nrmse_train = 0.10`).

```r
top_models <- OutputCollect$xDecompAgg %>%
  filter(rn == "(Intercept)") %>%
  select(solID, rsq_train, nrmse_train, decomp.rssd, mape) %>%
  arrange(desc(rsq_train))
knitr::kable(head(top_models), caption = "Top Candidate Models by R-squared")
```

Table 1: Top Candidate Models by R-squared

solID	rsq_train	nrmse_train	decomp.rssd	mape
151_96_1	0.6971556	0.1045021	0.3659704	0
139_97_1	0.6716268	0.1088176	0.3610319	0
99_49_1	0.6475716	0.1127329	0.3798754	0
44_99_1	0.6386621	0.1141490	0.3895422	0
126_93_1	0.6333399	0.1149866	0.3684264	0
63_85_1	0.6107305	0.1184788	0.3776532	0

Best Model Decomposition

After identifying the top-performing model based on R-squared and other evaluation metrics, contribution of each media channel is further examined using the selected model. The following analysis breaks down the **total sales impact attributed to each channel**, providing insight into their relative effectiveness and cost-efficiency. This forms the basis for ROI evaluation and future budget optimization.

```r
best_model <- "151_96_1"

model_contrib <- OutputCollect$xDecompAgg %>%
  filter(solID == best_model, rn != "(Intercept)") %>%
  select(
    channel = rn,
    coef,
    xDecompAgg,
    xDecompPerc,
    spend_share,
    effect_share)

knitr::kable(model_contrib, caption = "Channel-Level Contributions")
```

Table 2: Channel-Level Contributions

channel	coef	xDecompAgg	xDecompPerc	spend_share	effect_share
impressions_display	232.3137	10015.7735	0.4464750	0.4399985	0.6442488
impressions_social	148.6174	2569.3398	0.1145339	0.4616312	0.1652687
impressions_email	385.8309	2062.3165	0.0919323	0.0781407	0.1326553
impressions_direct	324.7267	899.0069	0.0400752	0.0202296	0.0578272

The table resents the channel-level decomposition of the best model (`solID = 151_96_1`), showing:

- **impressions_display:** Spend Share = 44.0%, Effect Share = 64.4%. Display is the most effective channel — contributing the most to sales relative to its share of impressions. This implies high ROI.

- **impressions_social:** Spend Share = 46.2%, Effect Share = 16.5%. Social ads received the most impressions but contributed relatively little to conversions — possibly inefficient or oversaturated.
- **impressions_email:** Spend Share = 7.8%, Effect Share = 13.3%. Email performed relatively well. Its small impression share shows it's very efficient and likely underinvested.
- **impressions_direct:** Spend Share = 2.0%, Effect Share = 5.8%. Direct impressions contributed some value despite a small share, suggesting opportunity for scaling with caution.

Channel-Level Contribution Analysis

```
model_contrib %>%
  mutate(roi = xDecompAgg / (spend_share * sum(xDecompAgg))) %>%
  select(channel, roi) %>%
  knitr::kable(caption = "Estimated ROI per Channel")
```

Table 3: Estimated ROI per Channel

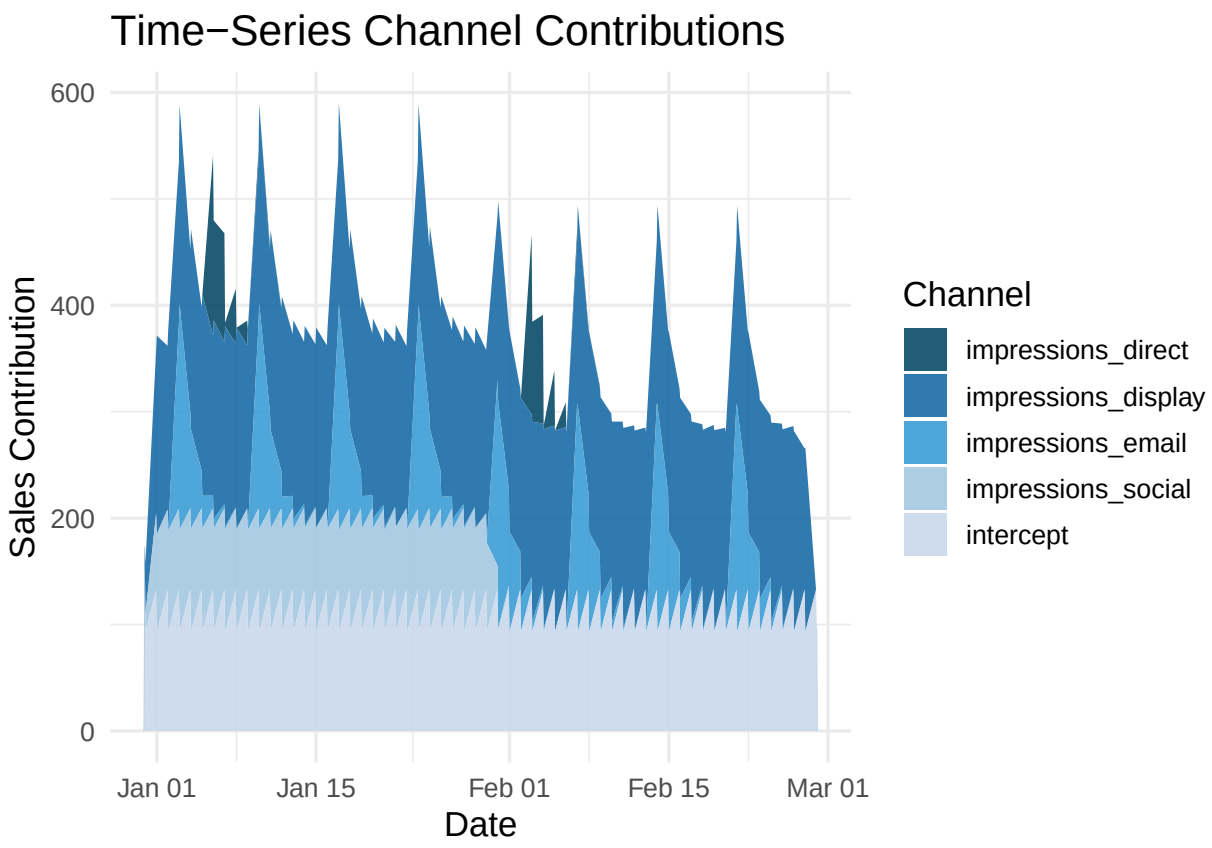
channel	roi
impressions_display	1.4642070
impressions_social	0.3580103
impressions_email	1.6976462
impressions_direct	2.8585401

To evaluate the cost-effectiveness of each marketing channel, Return on Investment (ROI) is calculated by dividing each channel's total contribution `xDecompAgg` by its share of total media spend. This metric provides a normalized view of how efficiently each channel converts media spend into sales.

Following this, a **time-series decomposition plot** is generated to visualize daily contributions of each media channel over the campaign period. The `xDecompVecCollect` object is reshaped into long format using `pivot_longer()`, enabling the use of `ggplot2` to create a stacked area chart.

```
selected_vec <- OutputCollect$xDecompVecCollect %>%
  select(-dep_var, -depVarHat, -solID, -cluster, -top_sol)

selected_vec_long <- pivot_longer(
  selected_vec,
  cols = -ds,
  names_to = "channel",
  values_to = "contribution"
)

blue_palette <- scales::hue_pal()(length(unique(selected_vec_long$channel)))
names(blue_palette) <- unique(selected_vec_long$channel)

blue_palette <- c(
  "#0A4C6A",
  "#1B6CA8",
  "#3B9DD5",
  "#A4C8E1",
  "#C9D9EC",
```

```
  "#7BAFD4"
)[1:length(unique(selected_vec_long$channel))]

ggplot(selected_vec_long, aes(x = ds, y = contribution, fill = channel)) +
  geom_area(position = "stack", alpha = 0.9) +
  scale_fill_manual(values = blue_palette) +
  labs(
    title = "Time-Series Channel Contributions",
    x = "Date", y = "Sales Contribution", fill = "Channel"
  ) +
  theme_minimal(base_size = 13)
```



This visualization helps illustrate how different channels drove sales over time and whether their impact was consistent or episodic. Together, these analyses offer both macro-level efficiency insights and micro-level temporal patterns, informing smarter budget allocation and campaign planning.

Interpretation:

- Display advertising yielded approximately **1.46 units of sales per unit of media spend**, indicating moderate efficiency.
- Social media advertising demonstrated **the lowest efficiency**, generating only **0.36 units of sales per unit of spend**, suggesting potential overinvestment or suboptimal performance.
- Email emerged as a **highly efficient channel**, delivering substantial sales impact relative to its relatively low share of total spend—indicating it may be underleveraged.

- **Direct impressions were the most efficient**, producing nearly three times the return in sales impact per unit of spend, signaling a strong opportunity for increased investment.

Robustness Check

To validate the reliability of the modeling insights, the top three performing models (ranked by R-squared and predictive accuracy) are selected for comparative analysis. By examining the **channel-level contributions** across these top models, the goal is to determine whether key findings (such as the dominance of display advertising or the high efficiency of direct impressions) remain consistent regardless of specific model variations.

This cross-model comparison helps ensure that conclusions are not overly dependent on a single model's parameters or random initialization. If the relative contribution patterns and ROI results are stable across the best-performing models, it strengthens confidence in the robustness of the insights and supports more confident media allocation recommendations.

```
top_ids <- top_models$solID[1:3]

multi_model_contrib <- OutputCollect$xDecompAgg %>%
  filter(solID %in% top_ids, rn != "(Intercept)") %>%
  select(solID, channel = rn, xDecompAgg, xDecompPerc, spend_share, effect_share)

knitr::kable(multi_model_contrib, caption = "Channel Contributions Across Top Models")
```

Table 4: Channel Contributions Across Top Models

solID	channel	xDecompAgg	xDecompPerc	spend_share	effect_share
99_49_1	impressions_display	9612.1445	0.4284823	0.4399985	0.6637131
99_49_1	impressions_social	2316.1134	0.1032458	0.4616312	0.1599263
99_49_1	impressions_email	1837.5681	0.0819136	0.0781407	0.1268830
99_49_1	impressions_direct	716.5538	0.0319420	0.0202296	0.0494776
139_97_1	impressions_display	8862.0751	0.3950464	0.4399985	0.6126833
139_97_1	impressions_social	2335.2081	0.1040970	0.4616312	0.1614456
139_97_1	impressions_email	2541.2058	0.1132798	0.0781407	0.1756873
139_97_1	impressions_direct	725.8769	0.0323575	0.0202296	0.0501838
151_96_1	impressions_display	10015.7735	0.4464750	0.4399985	0.6442488
151_96_1	impressions_social	2569.3398	0.1145339	0.4616312	0.1652687
151_96_1	impressions_email	2062.3165	0.0919323	0.0781407	0.1326553
151_96_1	impressions_direct	899.0069	0.0400752	0.0202296	0.0578272

Interpretation:

- Display ads consistently dominate, with the highest absolute and percent contributions (xDecompAgg and xDecompPerc) across all three models — and roughly aligned spend share (~44%). This suggests a well-justified investment.
- Social ads also have a stable spend share (~6%) but deliver a much smaller effect share (16%), indicating lower efficiency and potential overspend.
- Email consistently overperforms relative to its small spend share (~7.8%), delivering 12–17% of total effect, suggesting it's an underutilized high-ROI channel.

- Direct impressions, while minimal in spend (~2%), contribute around 5% of the total media effect, reinforcing their high efficiency.

The results show **a high level of consistency**, with display advertising emerging as the strongest contributor to sales, justifying its significant investment. Social media, however, accounts for the largest share of impressions but contributes relatively little to outcomes, suggesting overspending or inefficiency. Email and direct impressions consistently deliver high impact relative to their spend, making them strong candidates for increased investment. The alignment of findings across multiple top-performing models reinforces the robustness of these insights and supports confident, data-driven media optimization decisions.

Part IV: Conclusion

This **Marketing Mix Modeling (MMM)** analysis, powered by **Meta's Robyn** package, provided a robust framework to quantify and visualize the contribution of various paid media channels to overall conversions.

Model Performance

The best-performing candidate explained approximately **69.7% of the variance** in daily sales with relatively **low prediction error**, indicating a strong and reliable fit.

- **Top Performing Channels**
  - **Display advertising** and **email campaigns** were the most impactful drivers of revenue.
  - **Direct impressions** delivered the **highest ROI (2.86)** — showing exceptional efficiency despite their small share of total spend.
- **Underperforming Channel**
  - **Social media**, while receiving the highest share of impressions, showed significantly lower efficiency, with an ROI of just 0.36 — suggesting a key opportunity for budget optimization or reallocation.

Strategic Recommendations

Based on the model's decomposition and ROI analysis, it is evident that some marketing channels are significantly more cost-effective than others. To **improve overall return on investment (ROI)**, it is recommended to **reallocate budget away from underperforming channels**, particularly social media impressions, which demonstrated a low ROI of only 0.36 despite receiving a high share of spend.

In contrast, **email marketing and direct impressions** showed strong cost-efficiency and high impact relative to their budget allocation. Email achieved an ROI of 1.70 with a modest spend share, suggesting that it is currently underutilized. Direct impressions were the most efficient, yielding an ROI close to 3.0—making it a prime candidate for increased investment.

These following changes aim to optimize budget allocation, increase marketing efficiency, and ultimately drive higher incremental sales without increasing total spend.

- **Reduce spend on social media impressions** until ROI improves or targeting and creative strategies are refined.
- **Scale up investment in email campaigns**, especially automated or personalized outreach, which may yield even higher efficiency.

- **Expand direct marketing efforts**, potentially through retargeting, push notifications, or CRM-driven outreach, to capitalize on its superior cost-effectiveness.
- **A/B test new creatives or audience segments** within high-ROI channels to maximize marginal gains.

Limitations and Further Research

While this analysis provides a strong foundation for understanding media channel effectiveness using Meta's Robyn, there are several opportunities to enhance model accuracy and strategic applicability:

- **Seasonality Patterns:** The current model does not account for recurring trends such as day-of-week, month, or seasonal fluctuations. Incorporating seasonality would help distinguish organic sales cycles from media-driven impacts, leading to more precise attribution.
- **External Variables:** Key external factors such as macroeconomic indicators, competitor activities, weather events, or public holidays were not included in this iteration. These variables can influence consumer behavior independently of media spend and should be integrated to improve explanatory power and realism.
- **Scenario Simulations for Predictive Planning:** The analysis is currently retrospective. By leveraging Robyn's response curves and media saturation functions, future work can simulate "what-if" scenarios—such as reallocating budget across channels—to forecast potential outcomes and support forward-looking decision-making.
- **Computational Limitations:** Due to hardware constraints on a personal laptop, the model was trained with a reduced number of iterations (100), trials (200), and a single CPU core. These constraints limited the depth of the hyperparameter search and the total number of candidate models explored. Running the model on a cloud platform or high-performance machine could unlock more robust optimization and broader model discovery.