Faculty of
Mathematics
and Informatics

VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
MODELLING AND DATA ANALYSIS
MASTER'S STUDY PROGRAMME

Modelling return on advertising spend. Subscription-based B2C business case study

Master's thesis

Author: Armandas Budvytis
VU email address: armandas.budvytis@mif.stud.vu.lt
Supervisor: Dr. Andrius Buteikis

Vilnius

2024

Abstract

Applying the model form proposed by Google [3] a media mix model containing the carryover and saturation effects of marketing spend was estimated at the daily level on a dataset containing marketing spend and revenue for a period of two years. The models fit the data well and a strong carryover effect was found in the influencer channel, in particular, integrations with YouTube content creators, and a smaller one in the affiliates channel. The saturation effect estimation failed due to the poor identifiability of the function proposed for this task. In comparison to last-touch attribution, the model grossly underestimated the ROAS of the influencer channel.

Keywords: media mix modelling, bayesian methods, ordinary least squares, marketing

1 Introduction

Measuring the effectiveness of advertising campaigns is a question to which solutions have been developed for many years with the first mentions of marketing mix modelling as a proposed solution coming as early as in the 1960s[1]. The marketing landscape has evolved massively over the last couple of decades, in large part due to the dawn of digital media and, with it, digital advertising. The explosion of digital advertising resulted in many new advertising mediums becoming available for marketers. A few examples include ads on search engines, banners, video ads, sponsored influencer content, affiliate networks, etc.

The developments in technology and the rising tide of digitalization also gave birth to direct sales' attribution methods such as single-touch and multi-touch attribution which have become a popular option for companies to measure their online sales. Unfortunately, these solutions have drawbacks. Single-touch attribution is too simplistic, ignoring the possibility of the customer interacting with multiple sources of media before making a purchase. Multi-touch attribution can be a struggle to implement due to the difficulty in tracking unique user journeys across multiple platforms. On top of that, privacy is becoming more and more of a concern in the modern digitized world and third-party cookies are starting to be phased out as a result, making this even more difficult. Finally, both of these methods have no way of estimating anything related to offline campaigns.

Media mix models can help to circumvent these problems and are an area of quite heavy research, with even tech-giants such as Meta and Google developing and releasing their own frameworks for building such models. In this master thesis, I will be covering the relevant research in the field of marketing mix modelling, lending a larger part of focus on the Bayesian statistics based methods developed by Google.

Then, the proposed model by Google will then be adopted on a case of real-life data, which spans two years and comes from a B2C business which sells software on a subscription basis. This dataset also includes spend data from two, quite recently rising and previously mentioned, marketing mediums: affiliate networks and influencers (namely, YouTube content creators). The objectives of this thesis are to evaluate the proposed framework's viability for real-life applications where data is of a global scale (usually these types of models are estimated on a regional basis, for example, at the country level), as well as evaluate how the proposed model and framework are able to handle the fairly newly founded

influencer and affiliate channels.

Due to confidentiality, certain charts and values in tables will be scaled using a random constant.

2 Literature review

The purpose of this section is to define and provide literary background to metrics, transformations and modelling methods that are used in the modern digital marketing space in terms of profitability measurement. This section will cover return on advertising spend as a metric, the commonly used touch-based sales' attribution and media mix modelling methods and discuss the advantages and disadvantages of both of these methodologies.

Due to the scope of the thesis, the reasoning behind which is outlined in the introduction, more focus will lie with media mix modelling rather than touch-based attribution methods.

2.1 Return On Ad Spend (ROAS)

Return On Ad Spend (will be referred to as ROAS from this point on) is defined as the incremental revenue generated per dollar of expenditure on advertising [7]:

$$\text{ROAS} = \frac{\text{Revenue attributable to advertising (\$)}}{\text{Advertising spend (\$)}} \tag{1}$$

While advertising spend is fairly self-explanatory, revenue attributable to advertising warrants clarification. It is a significant challenge to adequately estimate the incremental effect that advertising has on a company's revenue. At its essence, accomplishing this feat is the main idea behind this thesis and the possible methods for this will be discussed in following sections.

Calculating advertising-driven revenue and, in turn, ROAS plays a crucial role in not only determining the success of completed and ongoing marketing ventures, but also strategic decisions with respect to both the long- and short-term. Even in startup environments with strong financial backing, heavy investment on marketing campaigns with low returns are unsustainable and understanding the way advertising impacts the bottom-line is of massive importance.

An important distinction is required here: while the two metrics are similar, ROAS is not equivalent to return on investment (ROI), which takes into account profit for the numerator in the fraction, rather than revenue. The formula for ROI can be seen below.[2]

$$\text{ROI} = \frac{\text{Net Profit(\$)}}{\text{Investment(\$)}} = \frac{\text{Revenue(\$)} - \text{Investment(\$)}}{\text{Investment(\$)}} \tag{2}$$

In essence, both metrics are measurements of profitability but ROAS is more commonly adopted by advertisers and it has become standard practice to speak in terms of ROAS rather than ROI in marketing contexts. As such, ROAS has established itself as one of the most commonly used Key Performance Indicators (KPI) for digital marketing campaigns with multiple digital marketing platforms allowing their users to optimize for it within their ecosystems like, for example, Google.

2.2 Touch-based attribution models

Touch-based attribution was born with the rise of digital advertising and the previously unavailable mediums for consumer journey tracking such as cookies and pixels. The main idea behind touch-based methods is that if we are able to get complete information about a customer's journey (for example, how many times he saw which ad and when), we can leverage that information and determine which touchpoints or, in other words, interactions with a company's marketing activities had the largest impact on the customer's conversion and, in turn, use this to optimize the conversion funnel and, as a consequence, optimize the ROAS.

Two different approaches exist for how a marketer can approach touch-based attribution modelling: rule-driven and data-driven. Rule-driven models operate on a advertiser-defined set of rules to determine the share of the conversion that each touchpoint was responsible for while data-driven models use statistical models trained on the collected data to estimate the impact. The following subsections will allow us to dive deeper and take a look at some examples as well as discuss some advantages and disadvantages.

2.2.1 Rule-driven touch-based attribution models

As briefly touched upon in the introduction to the "Touch-based attribution models" section, rule-driven models are based on predefined rules. In this subsection we will cover the most popular cases of this and explain what that means in more depth.

Last touch attribution model Likely the most simple and most popular model of this variation is the last touch attribution model. In this model, 100% of the conversion credit is assigned to the last marketing channel which a customer interacted with (that is possible to track and assign the contribution fact to, for example, via links with parameters or coupon codes)[10]. What this means is that, if, for example, a customer interacted with ten different marketing channels. In such a case, we'd rule that the last marketing channel that the user went through is solely responsible for the transaction. This method is easy to implement and interpret as you only need to be able to identify the channel behind the customer's session of purchase, but, unfortunately, its limitation is the assumption that only the final channel had any impact on the consumer's conversion.

Last non-direct touch attribution model This model is similar to the last touch attribution model, except it does not recognize the direct channel (entering the vendor's URL directly into the browser) as a valid channel which drives conversion. If direct is the last channel the customer goes through and buys, the credit for the conversion is assigned to the last non-direct marketing channel. If, for example, a customer interacted with ten marketing channels but the last channel was direct, the sale is attributed to the second to last channel, and if that was direct too, it continues to go further back to the first non-direct channel. This is where difficulty of implementation begins, as the marketer has to be able to identify not just the last interaction but also the second to last.

First touch attribution model First touch attribution is the polar opposite to the last touch attribution model: 100% of the conversion is attributed to the marketing channel with which the

consumer interacted first.

Linear attribution model This is the first model covered yet, which is a multi-touch attribution (MTA) model. Multi-touch attribution models are models who distribute the conversion credit across all of the contributing channels. The most simple model of this variation is the linear attribution model. The linear attribution model distributes the credit equally to each marketing channel that a customer interacted with, depending on the number of interactions. Building upon the previous example, if a customer had ten interactions with marketing media via nine different channels, the channel with two interactions gets 20% of the credit, while the rest of the channels get 10% each. The limitation is that the model assumes that each contributing channel has the same level of impact on the customer's conversion.

Time decay attribution Second in the line of credit distributing models is time decay attribution, in which the credit for the conversion is distributed in accordance with how recent the interaction with the marketing channel in question happened. The channels which interacted with the customer closest to the moment of conversion get the most credit, while the earliest channels get the smallest amount of credit. The limitations include this method using a constant rate of decay as well as the fact that upper-funnel marketing channels (the ones which focus on building brand awareness and building demand, educating on the use-case) getting underestimated due to being the "opening" channels more often than not.

Position based attribution Position based attribution tackles the inaccuracy between the worth of upper and lower-funnel channels . In this model, the channels of the first and last interactions get 40% of the credit each. The rest of the interactions, i.e. the 20% is distributed uniformly for each interaction. While this method attempts to assign equal credit to opening and closing channels, the limitation of this model is that synergistic effects are neglected and everything that happens in between the first and last interaction is, for the most part, ignored.

2.2.2 Data-driven touch-based attribution

Data-driven touch-based attribution methods take a different approach from the previously discussed rule-based models. Data-driven methodologies use statistical methods to estimate the impact of each customer interaction with marketing outlets.

While rule-based methods are fairly well-established and not experimented with as much anymore, data-driven methods are not as well-known, widespread and adopted, hence there is a lot more experimentation in this direction. In this subsection we will present and discuss a few examples of this approach.

Bagged logistic regression One proposed statistical solution for the marketing attribution problem involves bootstrapped logistic regression. [5] The authors define the advertising attribution question as a classification problem with a binary outcome variable. With that being said, as the authors of this solution note, human behaviour is complex, and data relating to this suffers from high correlation,

which often leads to multicollinearity and variable estimates with high variance. Not only this, but such situations cause strong variables to suppress weaker correlated variables [5]. This is why the authors employ the notion of standard deviation and take advantage of the fact that advertising campaign data often has a large sample size: a random subset of samples of both converting and non-converting users is obtained for both train and test datasets. A fixed ratio between converting and non-converting users is employed to address the dataset imbalance which is produced by the fact that a lot more consumers that are exposed to advertisements end up not converting rather than converting.

After subsetting the data, a multi-touch attribution (MTA) model is fit to the training data and the contribution of each advertisement channel is recorded. That same fitted model is then evaluated against unseen test data and the misclassification rate is also recorded. The process outlined above is repeated multiple times and the standard deviation of coefficient estimates is then obtained. The MTA model is evaluated on a basis of the average of standard deviations across all channels as well as the model's accuracy in predicting the outcome variable. In the ideal scenario, the model should have both stable coefficient estimates, i.e. small average standard deviation and strong predictive capabilities.

As for the MTA model, bagged logistic regression and a simple probabilistic model are proposed. Multiple models are considered due to the fact that advertising relies on a certain amount of subjectivity and multiple options allow the researcher the flexibility of choosing. In addition, having results from multiple models allows the researcher to cross-validate the general conclusions reached.

To focus more on the former option, logistic regression was selected due to the fact that logistic regression coefficients are highly interpretable, which would allow the advertisers to optimize their marketing efforts based on the coefficients as a result. While it is discussed that the bagging approach yields higher accuracy when combined with non-linear methods other than logistic regression, the bagging approach helps isolate collinearity[5]. As such, the common and interpretable logistic regression and bootstrap aggregation are combined, resulting in a model which achieves a stable and reproducible estimation. When evaluating the model via taking a look at coefficient variance and misclassification rate, it was found that while the misclassification rate was very similar to that of logistic regression, the bagged model achieved a much smaller variable variability, which is highly desirable given the attribution context.

As for the probabilistic model, it is based on first and second-order conditional probabilities. The model, in essence, is a second-order probability estimation and is generated by, firstly, computing the empirical probability of the main factors,

$$P(y|x_i) = \frac{N_{positive}(x_i)}{N_{positive}(x_i) + N_{negative}(x_i)} \tag{3}$$

and the pair-wise conditional probabilities for $i \neq j$.

$$P(y|x_i, x_j) = \frac{N_{positive}(x_i, x_j)}{N_{positive}(x_i, x_j) + N_{negative}(x_i, x_j)} \tag{4}$$

Here y is the binary outcome variable signifying conversion, $x_i, i, j = 1, ..., p$ denote p advertising channels, $N_{positive}(x_i)$ and $N_{negative}(x_i)$ are the number of converting and non-converting customers which interacted with channel i, respectively, and $N_{positive}(x_i, x_j)$ and $N_{negative}(x_i, x_j)$ are the number of converting and non-converting users exposed to both channels i and j.

Then, the contribution of channel i is computed at the converting customer level:

$$C(x_i) = p(y|x_i) + \frac{1}{2N_{j\neq i}} \sum_{j\neq i} (p(y|x_i, x_j) - p(y|x_i) - p(y|x_j)) \tag{5}$$

Here, $N_{j\neq i}$ is the total number of j's not equal to i, which in this case is $N-1$, or, in other words, the total number of channels minus one that a particular customer interacted with.

Second order interaction terms are included in the probability model due to the large amount of overlap between the impact of different touch points. Theoretically, the order may be even higher, but it was found that even in large datasets, the number of observations with the same third-order interactions drops quite strongly. Also, an important assumption is made in this model - the net effect of the second-order interaction is distributed equally across the two interacting channels.

It was found that the general conclusion between the two models was quite consistent but having multiple options allows both the flexibility of selecting the more appropriate model and cross-validating the results. Although, a drawback (the size of which largely depends on the advertiser's point of view), is that due to the problem being defined as a classification problem, rather than a regression problem (the conversion fact, rather than transaction revenue being used as the outcome variable), the model does not directly tackle the question of the monetary value that each channel brings.

While this is a detailed example of a possible solution, many alternative methodologies are available, such as deep learning solutions [9], game theory based methods[12] and methods utilizing Markov chains[4].

2.2.3 Media mix modelling

Media mix modelling has a different approach, even though the end goal and the advertisers' objective is the same. Media mix modelling takes marketing channel-level data, such as spend, impressions, clicks, emails sent, etc. as independent variables and total revenue or transactions as the outcome variable aggregated over time periods (usually at the weekly level) and uses them to estimate a regression model in order to derive the impact of each marketing channel on the bottom-line.

While touch-based attribution methods appeared with the rise of digital marketing and can only be used to measure digital advertising efforts, media mix modelling has been around in various forms since the 1960s and, naturally, also has the ability to quantify the impact of offline marketing channels, such as outside display, TV, radio, etc.

Google's Bayesian Framework One of the industry giants - Google has released a series of papers focusing on sale's attribution problems. While the advertising platforms that they run also employ the rule-driven attribution methods covered in section 2.2.1, their research and development is headed towards media mix modelling.

At the core of Google's proposed solutions, is a Bayesian approach of model estimation, allowing the advertiser to leverage prior knowledge (which we may, for example, derive from rules-driven touch-based attribution models). The proposed model also includes flexible functional forms to model the carryover (referring to marketing spend and activity having a delayed impact on sales) and shape (referring to the law of diminishing returns) effects of advertising.

To capture the carryover effect the time series of media spend for any given channel is transformed using the adstock function

$$\text{adstock}(x_{t-L+1,m}, ..., x_{t,m}; w_m, L) = \frac{\sum_{l_0}^{L-1} w_m(l) x_{t-l,m}}{\sum_{l_0}^{L-1} w_m(l)} \tag{6}$$

where $x_{t,m}$ is the spend of marketing channel m at time period t and w_m is a non-negative weight function. The cumulative media effect is then a weighted average of spend at time period t and previous $L-1$ time periods. L is the maximum duration of carryover effect assumed for the channel.

While different functional forms can be used, two in particular are suggested. The first one is geometric decay, which is commonly used for this purpose and is denoted as w_m^g.

$$w_m^g(l; \alpha_m) = \alpha_m^l \tag{7}$$

Here, $l = 0, ..., L-1$ and $0 < \alpha_m < 1$.

When using geometric decay as the weighting for the adstock function, an assumption is built-in that the effect of advertising is at its peak at the same time period as advertising exposure. This may not be the case with some marketing channels as the effect may take longer to build up. To account for this delay, the delayed weight function is introduced and denoted as w_m^d.

$$w_m^g(l; \alpha_m, \theta_m) = \alpha_m^{(l-\theta_m)^2} \tag{8}$$

Here, $l = 0, ..., L-1$, $0 < \alpha_m < 1$ and $0 \leq \theta_m \leq L-1$, which is the delay of the effect.

To capture the shape effect, the Hill function is used

$$\text{Hill}(x_{t,m}; \kappa_m, S_m) = \frac{1}{1 + (x_{t,m}/\kappa_m)^{-S_m}}, x_{t,m} \geq 0 \tag{9}$$

where $S_m > 0$ is the shape parameter also referred to as the slope, $\kappa_m > 0$ is the half saturation point due to $\text{Hill}(\kappa_m) = \frac{1}{2}$ for any value of κ_m and S_m. Also, as $x \longrightarrow \infty$, $\text{Hill}(x_{t,m}; \kappa_m, S_m) \longrightarrow 1$. Since it is required to allow different maximum effects for different marketing channels, the Hill function is multiplied with the corresponding regression coefficient β_m. The final shape transformation is then rewritten as:

$$\beta_m \text{Hill}(x_{t,m}; \kappa_m, S_m) = \beta_m - \frac{\kappa_m^{S_m} \beta_m}{x_{t,m}^{S_m} + \kappa_m^{S_m}} \tag{10}$$

Unfortunately, this proposed shape function is poorly identifiable and it is challenging to estimate the parameters well using statistical methods and using some examples it is possible to demonstrate that the individual parameters of this function are not estimated well, even though the curves are estimated well. Alternatives to this function are discussed but nothing definitive is said as it is out of the scope of the paper in question.

The two discussed effects can be combined in two different ways: either applying the adstock transformation to the media spend time-series and then applying the shape transformation, or doing it in reverse order. The advertiser has the freedom of selecting the order in which the transformations are applied but guidelines are presented: if the dataset which is being worked with displays a pattern

where the media spend in each time period is relatively small in comparison to the cumulative spend over time, it is preferable to apply the shape transformation after the adstock transformation, as the shape effect in each time period is not as obvious as that of cumulative media spend. In cases where the media spend is concentrated in isolated time periods and follows an on-and-off pattern, the reverse order of transformation should be followed. In the paper, the authors settle on the first option as it is more appropriate for the datasets they are working with.

In defining the model, an assumption about possible synergy between different media channels is also made. For simplification purposes, it is assumed that there is no significant synergy effect and the media effects are additive as a result. Finally, the response is modeled as

$$y_t = \tau + \sum_{m=1}^{M} \beta_m \text{Hill}(x^*_{t,m}; \kappa_m, S_m) + \sum_{c=1}^{C} \gamma_c z_{t,c} + \epsilon_t. \tag{11}$$

Here, y_t is the outcome variable at time period t, which can be sales or log transformed sales, $x^*_{t,m} = \text{adstock}(x_{t-L+1,m}, ..., x_{t,m}; w_m, L)$, τ is the baseline sales, γ_c is the effect of control variable z_c and ϵ_t is some white noise that is assumed to have no correlation with the model's other variables and has constant variance. It is also assumed that the relationship between the response and the control variables is linear.

As mentioned earlier, the Bayesian approach is used to estimate the model to allow the incorporation of prior knowledge into the model. The frequentist approach looks to find the maximum likelihood estimator

$$\hat{\Phi} = \arg\max_{\Phi} L(\mathbf{y}|\mathbf{X}, \mathbf{Z}, \Phi), \tag{12}$$

where Φ is the vector of parameters in model (11), $\mathbf{X}$ denotes all of the media variables, $\mathbf{Z}$ denotes all of the control variables, $\mathbf{y}$ is the vector of response values and $\text{L}(\mathbf{y}|\mathbf{X}, \mathbf{Z}, \Phi)$ is the log-likelihood given the data and model parameters.

On the contrary, when using Bayesian inference, the model parameters are treated as random variables. Bayesian inference is then based on the posterior distribution of the parameters given the data and the prior distribution $\pi(\Phi)$:

$$p(\Phi|\mathbf{y}, \mathbf{X}) \propto L(\mathbf{y}|\mathbf{X}, \mathbf{Z}, \Phi)\pi(\Phi) \tag{13}$$

When a single value is required as a summary result for media channels, the mean, median or mode of the posterior distribution are commonly used, while credible intervals can also be constructed using the posterior distribution, often using quantiles.

Gibbs sampling was implemented to sample from the specified model with the appropriate prior distributions, although the authors note other sampling methods can be implemented, like Hamiltonian Monte Carlo, for example.

The authors also note that the priors of the parameters have a large impact on the resulting posterior distributions. It is discussed that if the data is strongly informative, priors with the same support (support being defined as the set of values at which the probability density is positive) result in similar posterior distributions, while if the data is not as informative, the prior has a large impact

on the posterior distribution and sometimes, the posterior may even look almost identical to the prior.

Some comments about the choice of priors are also made. Retention rate α and delay parameter θ have constraints by definition, $[0, 1)$ and $[0, L-1]$, respectively. The retention rate can have even tighter constraints if there is strong prior information about the retention rate's prior knowledge, but, for the most part, it should have priors which are also define on $[0, 1)$ such as the beta or the uniform distributions. As for the delay parameter, uniform or a scaled beta distribution is suggested. For S - gamma distribution with a positive mode. For the regression coefficients β_m non-negative priors such as the normal distribution constrained to be non-negative is suggested, to work in the assumption that the effect of advertising cannot have a negative impact on sales. Finally, in the case of κ, as discussed earlier, the parameters of the Hill transformation become unidentifiable if this parameter is outside the range of observed advertising spend. In this case, we are unable to extrapolate the response beyond the observed media spend, but it can still be estimated well within the bounds of these observations if such a prior is defined for κ, that κ is constrained over the range of the observed media spend. This is shown via simulation studies in the paper.

Once the model's parameters are estimated, the model can be used to find the optimal channel mix which maximizes sales under a defined budget across a selected time period. The optimal setup $\mathbf{X^o} = \{x^o_{t,m}, t_0 \leq t \leq t_1, 1 \leq m \leq M\}$ for the time period (t_0, t_1) is found by

$$\max \sum_{t_0 \leq t \leq t_1 + L - 1} \hat{Y}_t(x_{t-L+1,m}, ..., x_{t,m}); 1 \leq m \leq M; \Phi) \tag{14}$$

with $\sum_{t_0 \leq t \leq t_1} \sum_{1 \leq m \leq M} x_{t,m} = C$. Here, C is the total budget of advertising spend in period (t_0, t_1) and $\hat{Y}_t$ denotes the predicted sales.

Two approaches are available to obtain the optimal mix within the Bayesian framework in this situation. The first option is to change the objective function (14) to

$$\max \frac{1}{J} \sum_{1 \leq j \leq J} \sum_{t_0 \leq t \leq t_1 + L - 1} \hat{Y}_t(x_{t-L+1,m}, ..., x_{t,m}; 1 \leq m \leq M; \Phi_j). \tag{15}$$

Here, Φ_j is the jth sample of the parameters and J is the total number of samples. Then, the constrained optimization problem is solved as usual and the optimal mix $\mathbf{X^o}$ is found. As for the second approach, each posterior sample of Φ is plugged into the objective function (14), which then is:

$$\max \sum_{t_0 \leq t \leq t_1 + L - 1} \hat{Y}_t(x_{t-L+1,m}, ..., x_{t,m}; 1 \leq m \leq M; \Phi_j). \tag{16}$$

Doing this, we acquire the best mix for the jth sample - $\mathbf{X^o_j}$.

The first approach leads to a more stable estimate of the optimal mix, and the second shows the variation of the estimated optimal mix. This variation is important as it informs the advertiser about how trustworthy the model is in it's allocation of the budget.

Using large scale simulation studies, the paper showed that the model can be estimated well on sufficiently large datasets, but biased estimates may be produced for more typical sample sizes, for example, two year of weekly-level data. In such cases, the defined priors have a substantial impact on the posterior distributions and differing priors may result in strongly different posteriors, which, in

turn, would lead to different attribution metrics. Therefore, it is of substantial importance to gather sufficient datasets and use informative priors. An example for how to gather informative priors is presented[8]: data of similar brands within one category may be pooled and the flat model (11) may be extended to a hierarchical Bayesian model, allowing for random effects of media and control variables for different brands. In addition to this, a hierarchical model employing geo-level data may be used[9]. In simulation, both of these approaches improve the performance of the model.

Furthermore, it was shown via the diagnostic analysis of residuals within an example, that the model in (11) may not capture the entirety of the autocorrelation of the response variable.

As far as academically documented media mix models go, this Google model together with its reach and frequency[11] and hierarchical[8] [6] extensions can be seen as the current state-of-the-art. While there is room for improvement, such as biased estimates when more standard sample sizes are used, there are options for addressing the issues as mentioned in previous paragraphs. Another technology giant - Meta, has been leading efforts in developing another media mix model framework called Robyn, but no academic publications have been made to this date.

3 Data and methodology

The subject of the research in this thesis is a B2C business, which provides subscription-based services to individual consumers. The dataset used in this thesis contains aggregated daily channel-level data of advertising spend and revenue from purchases via the company's website. Revenue is also distributed to each marketing channel, according to last-touch attribution based on clicks. Data availability spans two undisclosed years. The marketing strategy can be described as consistent always-on marketing and no longer-term experiments, which introduce large spending variance into the data were done over this timespan.

Certain external factors will be included, such as a dummy indicator for weekends (in the case of daily data), S&P 500 daily opening price, daily global average temperature and daily total global precipitation. These control variables should be sufficient to address weekly seasonality and economical and climate impact (in the climate variable case, it may be considered a yearly seasonality variable proxy if found significant). The explanations for all of the media variables can be found in the Appendix.

On top of all of that, a special interest of this research lies in finding out whether these methods are able to capture two newly and rapidly growing channels of the digital marketing space: affiliates and influencers.

Affiliate marketing works on commission for each acquisition, the partner gets a payout for each customer they bring or product they sell. This is quite different from standard, more traditional marketing channels, due to the fact that there is a direct coefficient via the payout between the sale of the product and the marketing spend of the channel. While the agreed-upon payout may be flat or set at a percentage from the revenue of the sale, meaning that the ROAS is almost entirely dependant on the threshold that the product owner decides is acceptable and, hence, we'd expect the channel to not conform to the saturation effect due to this, it is interesting to see whether a carryover effect exists. This may sound counterintuitive, but if we were to look at the spend as a proxy metric for how much traffic the affiliates manage to bring to the site or, better yet, the reach that they manage to

accumulate in that time period, it is very reasonable to expect that some part of the traffic may convert at a later time, especially considering the fact that different partners may employ different advertising tactics and methods.

On the other hand, influencers are generally paid for the content unit containing an advertisement on their social media. The data used in this thesis contains the expenditure on collaborations with influencers on YouTube (usually, a branded segment presenting and recommending the product) and the spend is registered at the date of the video release. This digital marketing approach is also quite new and not yet well explored. What makes the YouTube video integrations unique, is the fact that once the video is uploaded onto a YouTube content creator's channel, it's likely not going to get deleted. This means that potential customers may come across the video months (and even) years later by searching for a specific topic or via YouTube recommendations. Over time, the traffic on the video is, of course, diminishing, but as thousands of such videos get released every year, the effect can begin to accumulate. Not only that, but the YouTube algorithm may deem older videos relevant again at a random point in time, providing them with a new influx of traffic. Due to these reasons, we would expect a strong carryover effect for the influencer variable.

Since the available data (apart from the last-touch attribution sales' data) is aggregated over time, rather than customer-level journey data, media mix modelling is the technique that will be used for the main task of this research: delivering insights into each marketing channel's profitability and proposing solutions to optimize the channel mix. Previous experiments were done, regarding the training of data-driven multi-touch attribution models inside the company but the issue that was encountered was one of data availability: due to the nature of the company's marketing efforts spanning multiple advertising networks, sources and business models, the only data available which would allow to map the user journey in any way, came in the form of the medium which directed the user to start a session in the company's webpage via clicking a link. Unfortunately, if such a model were to be built, this disregards any type of impression data, leading to incomplete information and, in turn, a model which does not act as a valid representation of reality. Thus, last-touch attribution (coming from clicking a link or entering a coupon code assigned to a particular channel) is the main method in identifying the driving marketing forces behind sales up to now. This data, while not involved in the model development process, was used as a medium for comparing the results of the models developed in this paper to what is held true at the present day.

When faced with the creation and training of media mix models, there are multiple challenges. Firstly, while daily-level data is available, it is quite noisy, but, if we work with data at the weekly-level, we have barely the minimum recommended sample size for the building of these models. Furthermore, due to computation limitations, it may prove to be difficult to model carryover effects for longer periods in the daily-level data case. Secondly, marketing spend data can suffer from collinearity and multicollinearity. While this does not impact the accuracy of the estimation of the model, what we require from the model is trustworthy, stable and interpretable coefficient estimates, which collinearity and multicollinearity have a detrimental effect on. Thirdly, it is a question, whether the nature of the company's always-on marketing without the introduction of additional variance via strong spending adjustment experiments will allow for the discovery of the true ROAS estimates as well as the determining of the saturation and carryover effects. Another certain danger is present when variance is low,

which is that the intercept (which can be interpreted as momentum or sales unaccounted for) may make up a larger than is known to be true portion of the total sales.

3.1 Daily and weekly frequency data comparison

From what has been observed in multiple studies concerning media mix modelling, it appears that the preferred granularity of the data is weekly rather than daily. However, this does not mean that weekly data is to be considered as having the advantage over daily data by default. This section focuses on identifying the differences faced by choosing between daily and weekly data.

To achieve this, the data was looked at from multiple different angles. First of all, coefficient of variation (CV) was evaluated for the daily and weekly datasets' explanatory variables:

$$CV = \frac{s}{\bar{x}}, \tag{17}$$

Here, s is the sample standard deviation, and $\bar{x}$ is the sample mean.

This metric was used in order to compare the level of variance in relation to the mean at the different frequencies of data. In regression models, which focus on determining the effects of the explanatory variables, higher variance is desired as it leads to greater precision in coefficient estimates. The results can be seen in Table 1. Daily data has more variance in its independent variables and, in this sense, is preferred over weekly data in our particular case.

Channel	CV: Daily (%)	CV: Weekly (%)
Affiliates	26.21	21.92
Influencers	82.36	39.42
Branded Search Ads: Google	42.44	41.39
Google Discovery	89.56	80.81
Google Display	140.93	132.80
Nonbrand Search Ads: Google	30.03	27.31
YouTube Ads	191.89	160.89
Branded Search Ads: Bing	20.1	18.29
Nonbrand Search Ads: Bing	75.06	68.95
Temperature	10.02	10.11
Precipitation	5.67	4.46
S&P 500	6.42	6.47

Table 1: Coefficient of Variation

We also specified simple linear regression models to take a look at how well the spend data is able to be fitted on the weekly level data. In these models, all of the available variables are considered and insignificant variables are neither removed, nor merged into other related variables. In Table 2 it can be observed that there is a difference in how well the model is able to fit the target revenue data. On the other hand, the difference is not that substantial, R-squared of 0.892 is adequate.

Table 2 also contains the out-of-sample MAPE derived via predicting the data on the test dataset that was held out previously using the aforementioned daily and weekly models models. As we can see in both cases, it seems the models generalize quite well and don't show signs of overfitting.

Another important aspect that was touched upon very briefly earlier is collinearity and multi-

Data frequency	Linear regression R-squared	Out-of-sample MAPE
Daily	0.892	5.02%
Weekly	0.936	4.9%

Table 2: R-squared and out-of-sample MAPE for daily and weekly data

collinearity. As this is, to differing degrees, an everpresent question in media mix modelling, it is also an important aspect to evaluate when selecting the preferred data frequency. To estimate the degree of this issue present in the data, the Variance Inflation Factor (VIF) was calculated and is presented in Table 3 together with pairwise correlation coefficients which are visualised in Figure 1 and Figure 2.

Channel	VIF: Daily	VIF: Weekly
Affiliates	2.10	2.75
Influencers	1.06	1,32
Branded Search Ads: Google	2.65	3.49
Google Discovery	1.50	1.94
Google Display	1.52	1.74
Nonbrand Search Ads: Google	3.39	5.46
YouTube Ads	1.42	1.87
Branded Search Ads: Bing	1.47	1.65
Nonbrand Search Ads: Bing	1.91	1.65
Temperature	1.99	1.57
Precipitation	1.57	2.57
S&P 500	1.92	2.34

Table 3: Variance Inflation Factor. Initial estimations for daily and weekly data.

Using the multicollinearity rule of thumb of VIF < 5, only one variable fails to meet that criteria: "Nonbrand Search Ads: Google" in the weekly granularity dataset. Other than that, some multicollinearity is present but is not too severe in either case.

Having taken all of the above into account, it was decided to select the daily-level data for further work. From the evidence in this subsection, it can be said, that even though the media spend variables fit the target variable somewhat worse when regressed, the daily-level data exhibits signs of having more power and information when it comes to evaluating the explanatory variable coefficients as well as lower multicollinearity. In addition to this, there is a large advantage in sample size, in total (training and testing sets) daily data has 730 observations, while weekly data has 104.

Before beginning the modelling phase, each column containing continuous data was transformed using min-max normalization

$$x'_{i,j} = \frac{x_{i,j} - \min(x_i)}{\max(x_i) - \min(x_i)}. \tag{18}$$

Here, $x_{i,j}$ is the jth observation of column i in the dataset containing the explanatory variables, and $\min(x_i)$ and $\max(x_i)$ are the minimum and maximum variables in that column, respectively.

This, effectively, scales the data down into a range of (0,1). While this has no bearing on the results of the OLS model specification, except for the difference in coefficient interpretation (model goodness-of-fit remains at the same level), data that is scaled makes it easier for Bayesian methods to converge

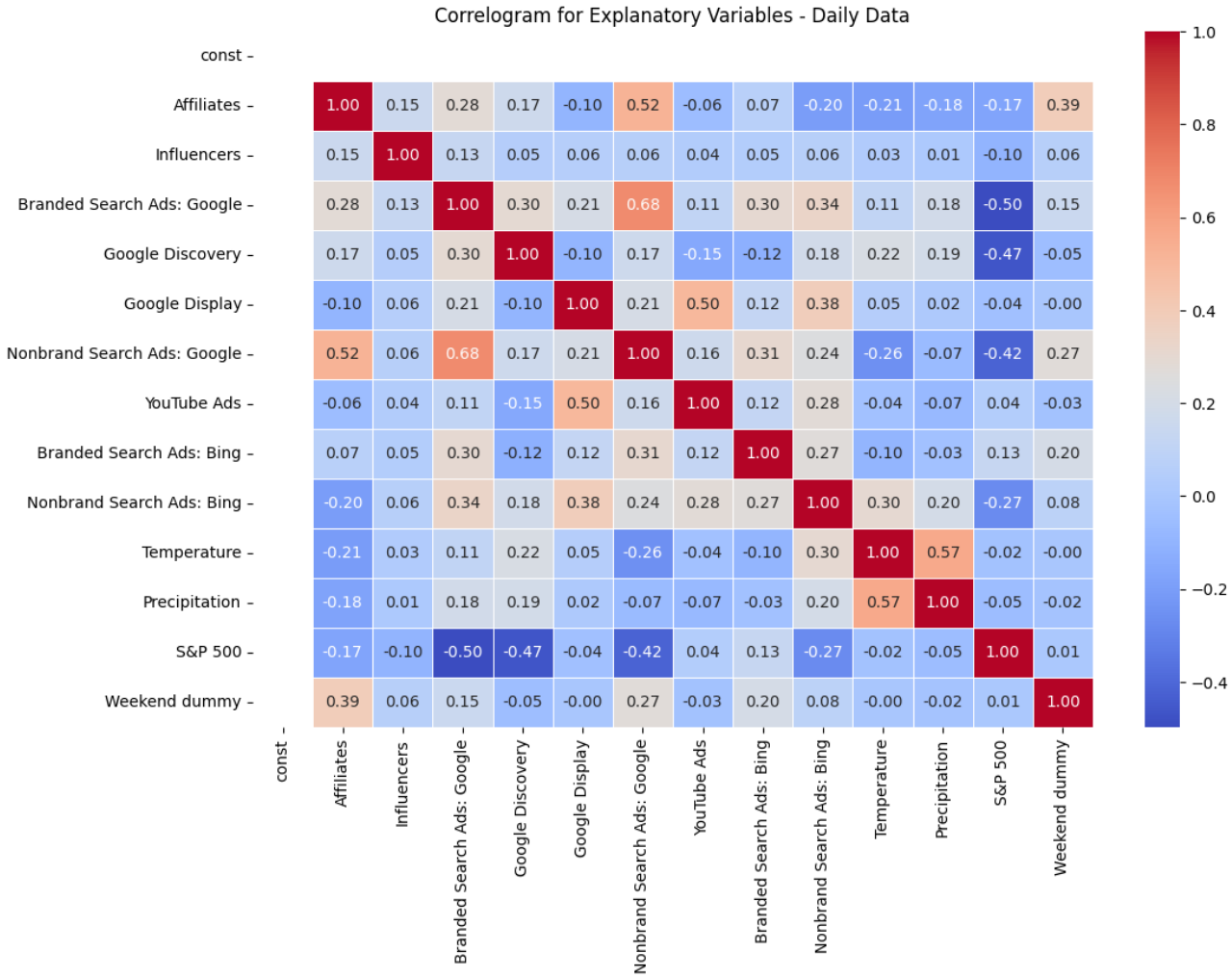

Figure 1: Correlogram for Explanatory Variables - Daily Data

and allows for quicker sampling speeds. The modeller also benefits as setting the priors becomes more intuitive when the coefficient priors are contained within more standard bounds.

3.2 Model specification

Two different methods for media mix model estimation were used. This allowed us to not only select the best methodology for the given case but also cross-reference the results as we should be expecting a certain level of consistency across the different methods.

Firstly, a linear regression model was estimated as the baseline model and treated as the base solution. The OLS model was also used to address insignificant variables by either removing them from the model, or combining them with another correlated variable into a subcategory.

The model's goodness-of-fit was then evaluated using R-squared, and out-of-sample MAPE (after scaling back to the original scale).

Once the final results from the OLS estimation were obtained, the remaining significant variables are saved and further used in the Bayesian framework.

The Bayesian framework containing saturation and carryover effects is described in detail in section

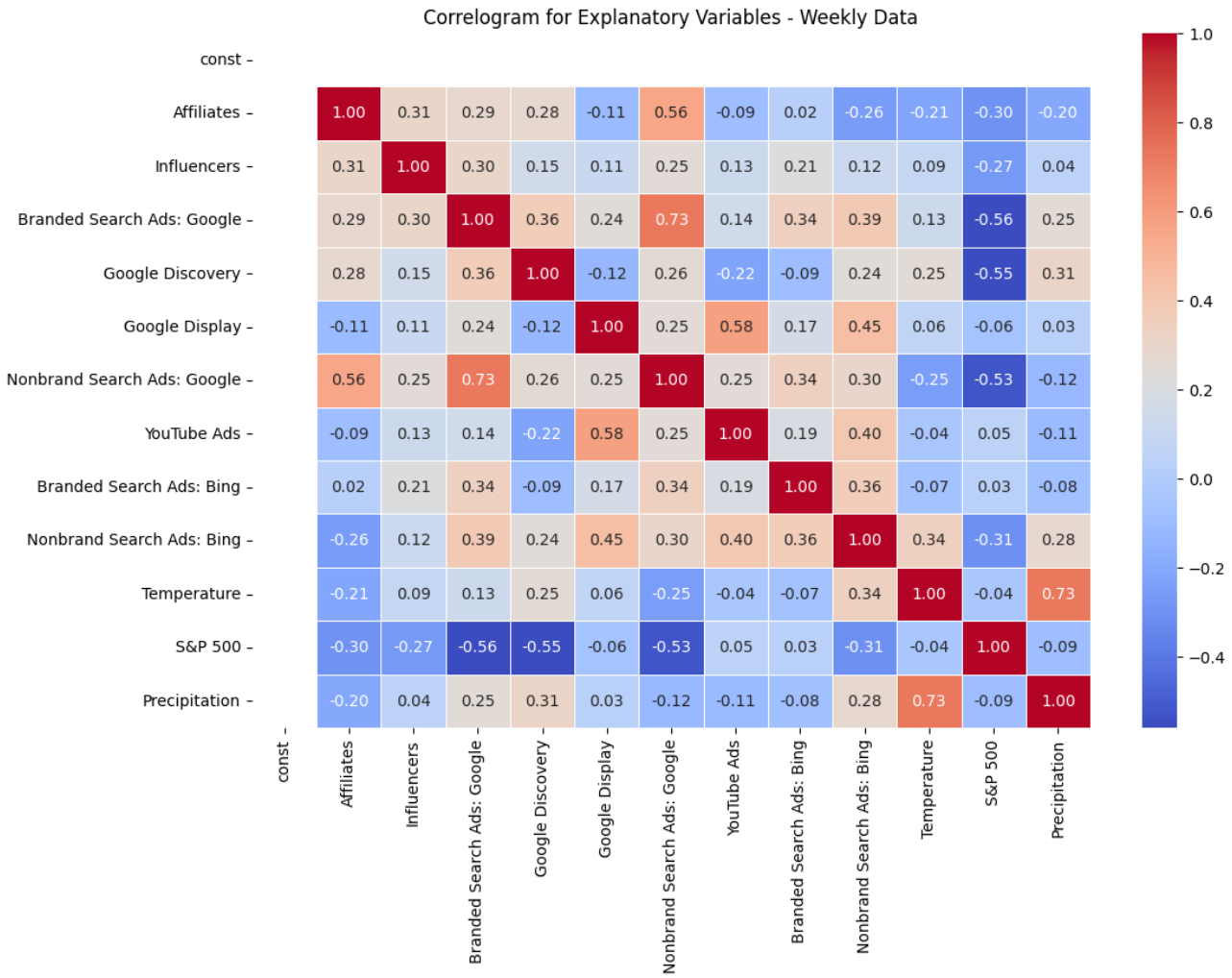

Figure 2: Correlogram for Explanatory Variables - Weekly Data

2.2.3. To accomplish this, the PyMC environment in Python was used to perform No-U-Turn Sampling (NUTS).

PyMC allows the user to specify a probabilistic model for the relationship between the explanatory variables and the response variable. The specification is done using the definition of the likelihood function (which in our case is equation 11), necessary parameters and their priors. It was determined to adopt the Hill function to capture the saturation effect and geometric decay to capture the carryover effect for the media variables. For the control variables, the parameter γ_m was specified. As for the media variables, four parameters were specified for each of the media channels: β_m as the coefficient for the scaled ROAS, κ_m and S_m for the Hill function and α_m for the geometric decay weight. Due to demanding computational capabilities, the maximum length of the carryover effect was set at 14 periods.

In regards to prior information, for the β_m coefficient priors for each channel we will be looking to use truncated normal distributions truncating the lower tail at zero, setting the μ_m parameter to the mean of the scaled ROAS of channel m, according to the previously specified OLS model. The σ_m^2 parameter will also be taken from the coefficient estimate from the linear regression model. The same

goes for the control variables and intercept.

In terms of the truncated normal distribution, the probability density function can be defined as

$$f(x;\mu,\sigma,a,b) = \frac{1}{\sigma}\frac{\phi(\frac{x-\mu}{\sigma})}{\Phi(\frac{b-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})} \tag{19}$$

for $a \leq x \leq b$ and $f = 0$ otherwise. Here,

$$\phi(\frac{x-\mu}{\sigma}) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2) \tag{20}$$

is the probability density function of the standard normal distribution and

$$\Phi(x) = \frac{1}{2}(1 + \text{erf}(x/\sqrt{2})) \tag{21}$$

is the standard normal distribution's cumulative distribution function.

Regarding the priors of other variables, the prior for α_m is uninformative and was set as the uniform distribution over it's entire acceptable range. As for κ_m and S_m, they have weakly informative priors. κ_m is set to not be larger than the maximum observed value of the media variable x_m as when it exceeds that bound, the βHill function becomes unidentifiable; S_m is set to the selected range, because when it leaves these bounds, the Hill function obtains an undesirable shape, that is not indicative of the saturation effect a marketing spend variable may have. The bounds for each of which are presented in Table 4.

Parameter	Distribution limits
κ_m	$(0.01;\ \max(x_m))$
S_m	$(0.5;\ 3)$
α_m	$(0.01;\ 0.99)$

Table 4: Uninformative priors for the media channel parameters

Once the sampling and estimation was complete, the posterior was then sampled and a posterior predictive check was done. Additionally, pointwise mean estimates of the parameters were taken and used to evaluate the model fit on the train and test datasets.

4 Results

The results from estimating an OLS model on all of the explanatory variables in the dataset can be found in Table 5. The coefficients are not representative as after the min-max normalization the data was also scaled by a random constant for each channel to preserve confidential information while preserving model fit.

Step-by-step removal and grouping of insignificant variables was then performed. Some variables which were insignificant or had pairwise correlation with similar variables were grouped (their values were summed) into logical clusters, paying respect to the type of advertisement that they describe. Due to the sample size and the specification of the transformations, continuous sampling will require exponentially large computational efforts and removing insignificant variables (p-value exceeding 0.1)

Variable	Coefficient	Std. dev	t	P >|t|
Intercept	0.0122	0.012	1.011	0.312
Affiliates	8.2800	0.200	41.386	0.000
Influencers	0.4445	0.128	3.478	0.001
Branded Search Ads: Google	-0.0213	0.079	-0.268	0.789
Google Discovery	-0.0186	0.014	-1.349	0.178
Google Display	-0.0337	0.047	-0.710	0.478
Nonbrand Search Ads: Google	0.8023	0.139	5.771	0.000
YouTube Ads	0.0114	0.063	0.182	0.856
Branded Search Ads: Bing	0.0111	0.031	0.362	0.717
Nonbrand Search Ads: Bing	0.0459	0.028	1.651	0.099
Temperature	-1.2772	0.162	-7.898	0.000
Precipitation	-0.0131	0.016	-0.842	0.400
S&P 500	0.1084	0.027	4.010	0.000
Weekend dummy	-0.0339	0.026	-1.304	0.193

Table 5: Initial OLS estimates (Min-max normalized data times a random constant for anonymization)

Variable	Coef	Std. dev	t	P>|t|	[0.025	0.975]
Intercept	0.0177	0.010	1.796	0.073	-0.002	0.037
Influencers	0.4087	0.127	3.214	0.001	0.159	0.658
Paid Search	2.2428	0.303	7.413	0.000	1.649	2.837
Affiliates	8.2703	0.161	51.394	0.000	7.954	8.586
Banners and YouTube	0.0566	0.107	0.530	0.596	-0.153	0.266
Temperature	-0.5434	0.042	-12.900	0.000	-0.626	-0.461
S&P 500	0.2844	0.053	5.405	0.000	0.181	0.388

Table 6: Final OLS model coefficients. Min-max and random scaling applied.

will assist us in optimizing the sampling computations as well, and not just simplifying the OLS model.

The final OLS model can be inspected in table 6. In this iteration new random constants were applied to Paid Search and Banners and YouTube variables rather than summing old values. This will no longer come up in the rest of the thesis.

It is also worthy to note, that while the p-value for Banners and YouTube exceeds 0.01, it was decided to keep it in the model as it may have carryover effects that make it significant.

In terms, of model fit, the model's R-squared is equal to 0.889, which is quite good. Also, the MAPE was calculated after scaling the dataset back to its original values. It was found that the MAPE on the train set is 4.2%, which is adequate. Figure 3 shows the OLS model's fit on the train dataset. ROAS coefficient analysis is left for later, and we'll be moving to discuss the Bayesian modelling results.

Firstly, a linear regression was estimated using the Bayesian regression methods and it returned the same results as OLS, which signifies no issue in the initial method. Following that, an attempt was made at estimating all of the parameters together for the regression model containing carryover and saturation effects. Unfortunately, even after several respecifications, experiments with priors, PyMC hyperparameters, samplers, etc. it resulted in divergences or unreasonable computation times. Hence, it was required to alter the approach and it led to an additional assumption being built into the model.

The approach that was settled on was modelling the carryover and saturation effects in steps.

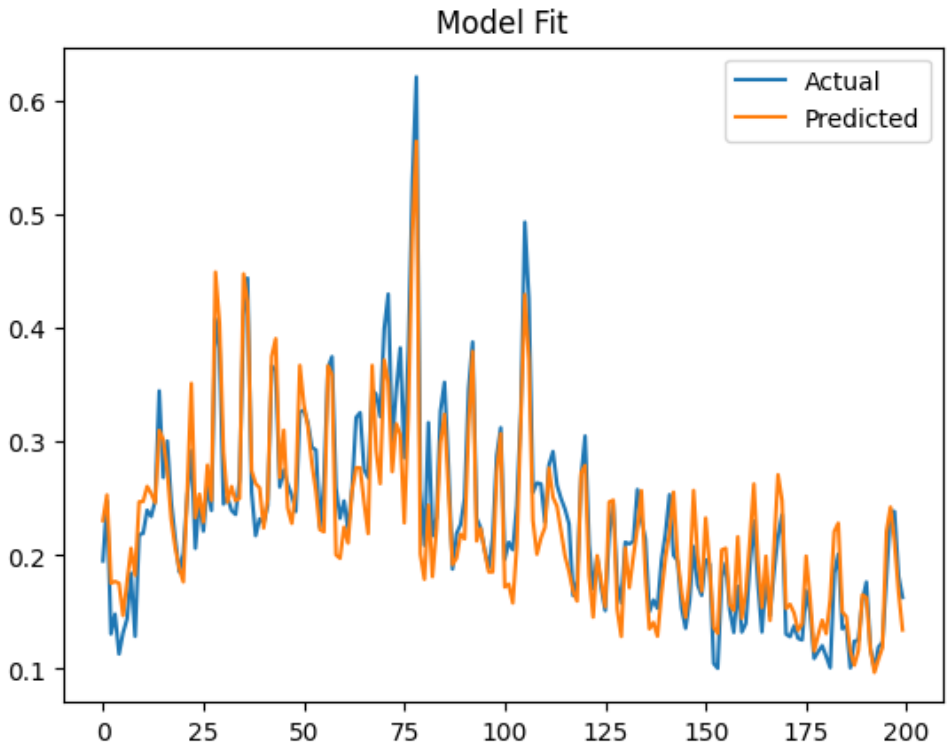

Figure 3: OLS performance against train set. 200 observations.

First, building a model that only included the carryover effect and then, building a model that did not optimize the carryover anymore but rather, just used the mean of the parameter derived from the previous model as a constant (if the carryover effect in the previous iteration was deemed significant and identified, i.e. converged to a specific value). While the saturation and carryover transformations should not have an effect on one another, we still have to treat this as an information loss due to the fact we are effectively assuming that the decay weight for the carryover is a constant in the final model.

The trace plot and summary for the model with only the carryover effects accounted for can be found in Figure 4 and Table 7 respectively. Note that Banners and YouTube variable is not present as it was insignificant even with carryover effects accounted for. The intercept term and the carryover effect for the Paid Search was also removed due to being insignificant. It is worthy to note that the estimates appear to be very stable judging by the coefficient standard deviations and the Highest Density Intervals, an important feature of a stable media mix model.

As we can see from the table and the posterior plot, the remaining significant parameters appear to have converged nicely.

Taking the pointwise mean estimates of the parameters we can evaluate the model fit. Figure 5 shows the fit on the train set while figure y shows the fit when the test set is predicted. As for the diagnostic numbers: R-squared is 0.907, an improvement over the linear model, MAPE for the train set is 4.3%.

Taking a look at the carryover coefficients for influencers and affiliates we can note that the Influencer channel has a strong carryover effect associated with it, decaying at a slow rate in comparison to the other two variables, which was to be expected. Meanwhile, the affiliates channel also enjoys a carryover

Parameter	Mean	Std. dev	HDI interval 3%	HDI interval 97%	ESS of mean	ESS of st. dev	$\hat{r}$
gamma_Temperature	-0.579	0.028	-0.633	-0.528	16689.0	13701.0	1.0
gamma_S&P 500	0.283	0.027	0.231	0.332	16068.0	13643.0	1.0
beta_Influencers	0.391	0.059	0.279	0.500	11713.0	9005.0	1.0
carryover_Influencers	0.871	0.032	0.811	0.933	11584.0	8703.0	1.0
beta_Affiliates	7.852	0.123	7.619	8.084	13255.0	13339.0	1.0
carryover_Affiliates	0.075	0.019	0.040	0.111	11500.0	8074.0	1.0
beta_Paid Search	1.669	0.159	1.371	1.966	14432.0	11354.0	1.0
sigma	0.035	0.001	0.033	0.037	16035.0	14417.0	1.0

Table 7: Summary of model containing the carryover effect without saturation applied

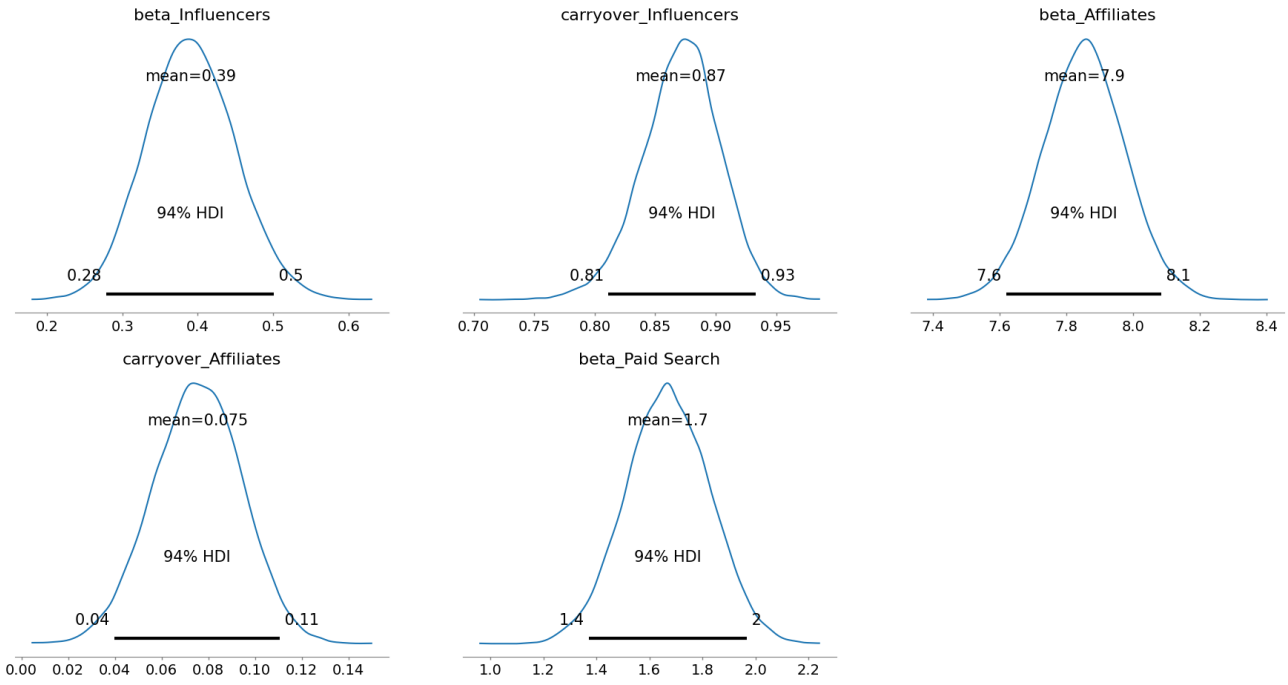

Figure 4: Posterior plot of the carryover model

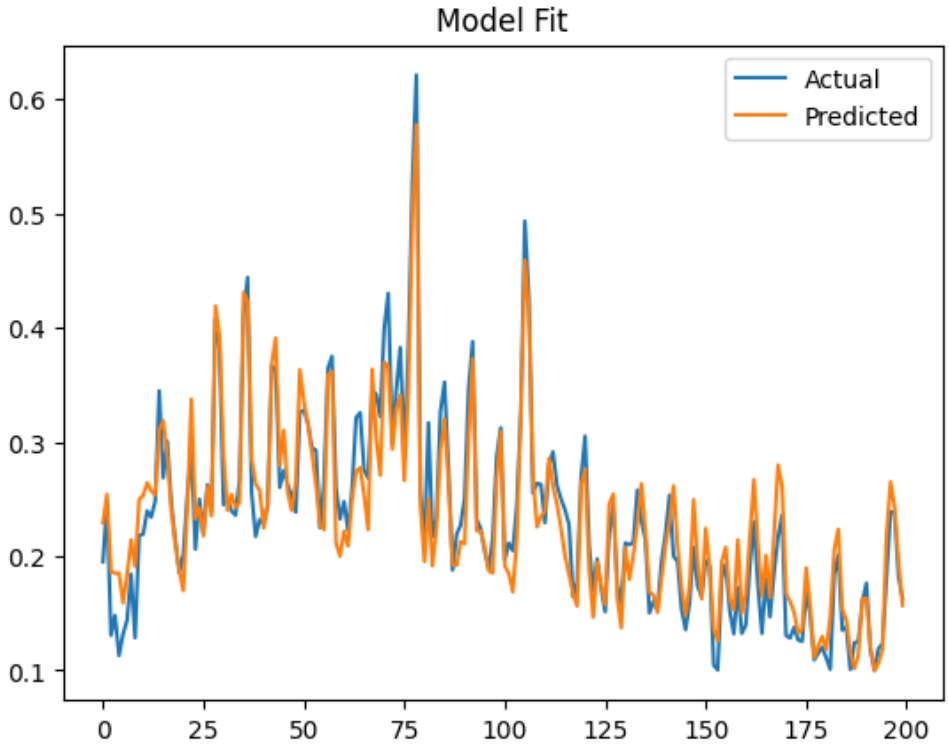

Figure 5: Carryover effect model fit on train dataset. 200 observations.

effect but of a much smaller scale. Little to no carryover effect is to be expected keeping mind the model under which the channel operates, but as affiliate marketers choose their own methods for promoting the product, it is reasonable to expect that the effect would expand beyond that one day, if we look at the marketing spend as a proxy of the marketing reach.

Having acquired the estimates for the carryover effects, the third model with saturation effects was then estimated (results in Figure 6 in the Appendix). Unfortunately, it was required to reject this model due to the fact that it does not provide additional information because for all of the channels, the κ converge well beyond the max values of their respective spend variables. This, makes the βHill function unidentifiable[2]. This can be observed in the Figure 6 which can be found in the Appendix.

Removing the min-max scaling but keeping the random scaling for anonymization we arrive at the ROAS estimates in Table 10 which can be found in the appendix.

It can be observed how large of an impact the estimation of the carryover effect had in comparison to the original OLS computation and how media mix models assign credit differently in comparison to last touch attribution overall. Not only that, but the impact of Paid Search reduced drastically when using media mix modelling methods in comparison to last-touch attribution - with carryover effects accounted for it decreased 3 times. From the perspective of the marketing domain, it makes sense due to the reasons discussed earlier, in particular about the influencers channel and its long term effects. While the affiliates channel benefited marginally from the minor carryover effect, it can be seen that it was also underestimated by the last-touch model, when comparing with our media mix modelling methods, by around one third.

Nevertheless, there is still quite a huge gap to be addressed in terms of influencer channel impact

estimation. With the way the channel works, last touch attribution should be missing out on a substantial number of transactions and indirect impact due to the scale of the influencer channel in this particular case. But from our results we can see that the last-touch attribution estimate is still around 4x bigger. Even if the hypothesis about the influencer channel driving sales indirectly (when speaking in last touch attribution terms), the gap between the estimates is far too large to be representative of the truth.

5 Conclusion and considerations

In this case study, two media mix modelling solutions were applied, firstly, the classical OLS model and, secondly, the Bayesian framework with the potential for estimating carryover and diminishing returns (saturation) effects. Models of both frameworks fit the data well with the Bayesian model having a slightly better fit, even though the diminishing returns effects turned out to be unidentifiable and only carryover was estimated. The results from these two models were then compared to ROAS estimates derived via last-touch attribution, a popular and cheap method of assigning sales to marketing channels.

It was found that, when comparing to last-touch attribution, the impact of paid search ads was 4x smaller when using estimates from the models. Furthermore, the affiliate channel's impact was one third bigger in the marketing mix models than in last touch attribution. For these two channels, the truth may lie somewhere inbetween.

It was hypothesised that the influencers channel, which is made up of YouTube content creators with which the company do collaborations, has a strong lasting effect across time when videos are released. Unfortunately, the choice of daily data may have been detrimental to coming closer to determining the true impact of the influencers channel in this case. Limitations arise when applying functions that specify carryover effects over long periods on daily data within the current available Bayesian sampling framework: computational time costs grow exponentially and the necessity arises to settle on a lower window for estimating the carryover effects. In the case of this study, the maximum time that marketing spend could've had an impact on future revenue was 14 days. This resulted in a 4x smaller influencer ROAS than is estimated via last touch attribution, while we would expect it to be above this metric, if a robust model was available due to the intricacies and scale of the channel. The affiliates channel also showed a light carryover effect.

In addition to this, it must be noted that estimating the saturation of the respective channels proved challenging. Half-saturation parameter κ converged to larger than the maximum observed values for the media parameters, which led to the βHill function becoming unidentifiable. More work needs to be done to find other functions which may be able to fulfill this modelling need while having better identification properties.

In regards of this particular study, there are still multiple things that can be done from the company's side to include future iterations of this modelling problem. For example, implementing the Geo level hierarchical model [6]. Another useful exercise would be independent channel experimentation, where channels would experiment with their spend numbers across predetermined independent time periods should result in additional information in the data, which the models should then be able to capture

and provide more reliable estimates. A model based on weekly data should also be experimented with but this may require that the data cover a longer time period, especially if more granular marketing channels were to be analysed.

References

[1] Neil H. Borden. The concept of the marketing mix. *Journal of Advertising Research*, 4:2–7, 1964.

[2] Paul W. Farris, Neil T. Bendle, Phillip E. Pfeifer, and David J. Reibstein. *Marketing Metrics: The Definitive Guide to Measuring Marketing Performance*. Pearson, 2016.

[3] Yuxue Jin, Yueqing Wang, Yunting Sun, David Chan, and Jim Koehler. Bayesian methods for media mix modeling with carryover and shape effects. *Google Inc.*, April 2017.

[4] Lukáš Kakalejčík, Martina Ferencova, Paulo Angelo, and Jozef Bucko. Multichannel marketing attribution using markov chains. *Statistika Statistics and Economy Journal*, 101(2), January 2018. Affiliations: GrowthPro, University of Presov in Presov, University of Brasília, Technical University of Kosice - Technicka univerzita v Kosiciach.

[5] Xiaofei Shao and Liang Li. Data-driven multi-touch attribution models. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1269–1277. ACM, 2011.

[6] Yunting Sun, Yueqing Wang, Yuxue Jin, David Chan, and Jim Koehler. Geo-level bayesian hierarchical media mix modeling. *Google Inc.*, 2022.

[7] Jon Vaver and Jim Koehler. Measuring ad effectiveness using geo experiments. *Google Inc.*, 2019.

[8] Yueqing Wang, Yuxue Jin, Yunting Sun, David Chan, and Jim Koehler. A hierarchical bayesian approach to improve media mix models using category data. *Google Inc.*, April 2017.

[9] Dongdong Yang, Kevin Dyer, and Senzhang Wang. Interpretable deep learning model for online multi-touch attribution. 3 2020.

[10] C. B. Yuvaraj, B. R. Chandavarkar, V. S. Kumar, and B. S. Sandeep. Enhanced last-touch interaction attribution model in online advertising. 2018.

[11] Yingxiang Zhang, Mike Wurm, Alexander Wakim, Hongyu Li, and Ying Liu. Bayesian hierarchical media mix model incorporating reach and frequency data. *Google LLC.*, July 2023.

[12] Kaifeng Zhao, Seyed Hanif Mahboobi, and Saeed R. Bagheri. Shapley value methods for attribution modeling in online advertising. *Data and Analytics R&D, GroupM*, 2018. Emails: Kaifeng.zhao@groupm.com, Seyed.Mahboobi@groupm.com.

Appendix A

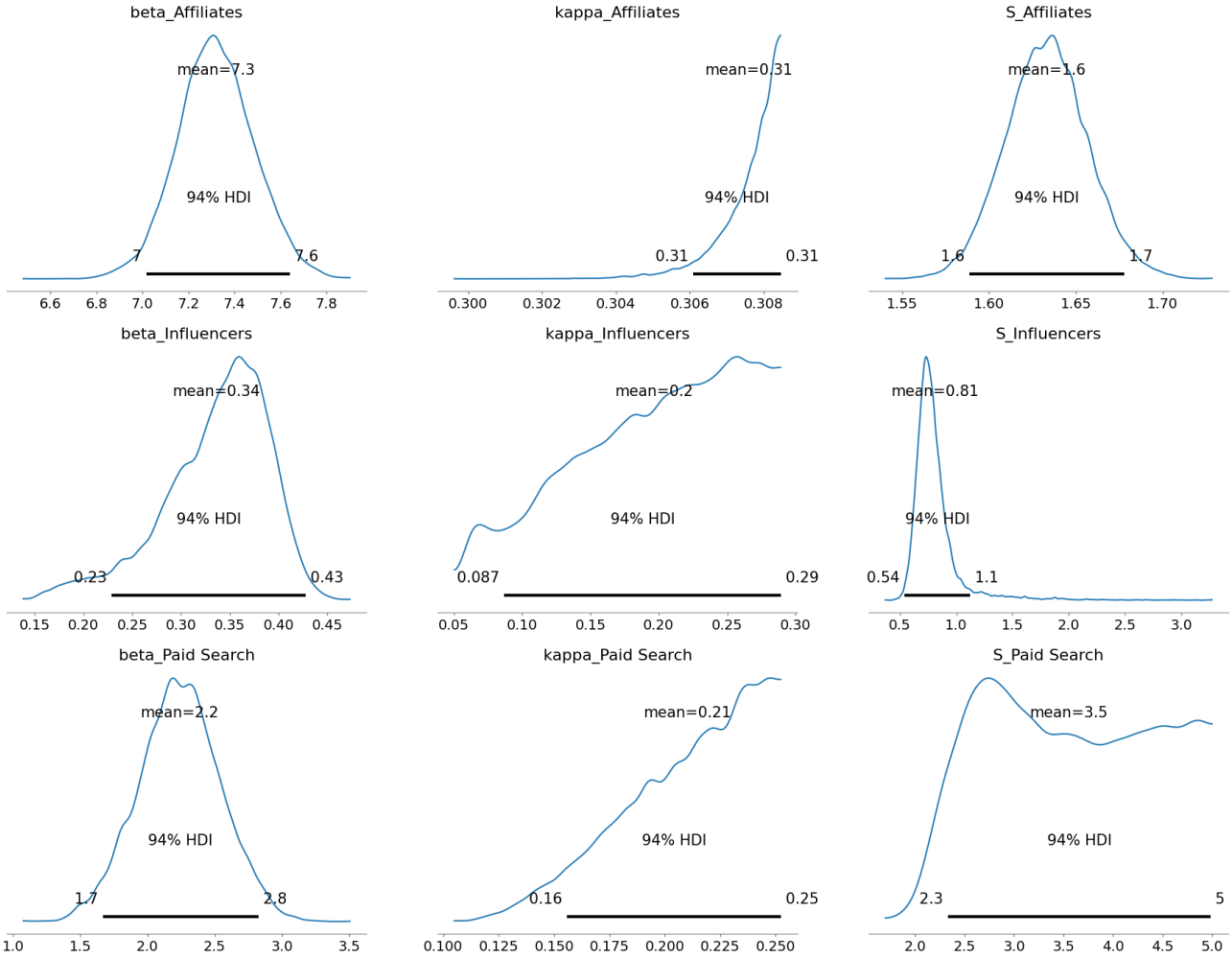

Figure 6: Saturation effects model posterior distributions

Variable	Explanation
Affiliates	Total amount of payouts to affiliates who partner with us and get paid for each acquisition
Influencers	Total amount of commissions paid for the content units, containing promotion, that our Influencer partners release
Branded Search Ads: Google	Spend on ads displayed when users search for the company's brand on Google.
Nonbrand Search Ads: Google	Spend on ads shown for generic queries, not specific to a brand on Google.
Branded Search Ads: Bing	Spend on ads displayed when users search for the company's brand on Bing.
Nonbrand Search Ads: Bing	Spend on ads shown for generic queries, not specific to a brand on Bing.
Google Discovery	Spend on ads shown across Google's discovery feed and platforms.
Google Display	Spend on visual ads on websites who have diverse formats: text, image, or interactive.
YouTube Ads	Spend on video ads on YouTube.

Table 8: Ungrouped media variable explanation

Variable	Contains	Explanation
Affiliates	Affiliates	Total amount of payouts to affiliates who partner with the company and get paid for each acquisition.
Influencers	Influencers	Total amount of commissions paid for the content units, containing promotion, that the company's Influencer partners release.
Paid Search	Branded Search Ads: Google, Branded Search Ads: Bing, Nonbrand Search Ads: Google, Nonbrand Search Ads: Bing	Total spend on advertisements in search results. Paid for each click.
Banners and YouTube	Google Discovery, Google Display, YouTube Ads	Total spend on ads on YouTube, Google Discovery and Google Display. Paid for each click.

Table 9: Grouped by summing the values (final) media variables

Channel	Last-touch attribution model ROAS estimate	OLS ROAS estimate	Carryover effects model ROAS estimate	Last-touch vs OLS	Last-touch vs Carryover	OLS vs Carryover
Affiliates	9.59	13.78	14.12	69.59%	67.91%	97.59%
Influencers	1.78	0.07	0.44	2542.86%	404.54%	15.91%
Paid Search	37.157	16.02	11.40	140.52%	325.94%	140.53%

Table 10: ROAS estimate comparison of different methods

Appendix B

This appendix contains the Python code created while working on this thesis.

```
# The spend and revenue data was obtained via SQL joins of many differnent sources and will not be sh

import pandas as pd
import requests

# Downloading temperature data
url = "https://climatereanalyzer.org/clim/t2_daily/json/era5_world_t2_day.json"
response = requests.get(url)
data = response.json()

years = ['2022', '2023']
filtered_data = [entry for entry in data if entry['name'] in years]

daily_data = []
for entry in filtered_data:
    year = int(entry['name'])
    daily_data.extend([(year, date, temperature) for date, temperature in enumerate(entry['data'], st


columns = ['Year', 'Day', 'temperature']
df = pd.DataFrame(daily_data, columns=columns)

df['date'] = pd.to_datetime(df['Year'].astype(str) + df['Day'].astype(str), format='%Y%j')

df = df.drop(['Year', 'Day'], axis=1)
df = df.set_index("date")

df
df.to_csv('data/world_avg_temperature.csv', index=True)

# Acquire precipitation data

import cdsapi

c = cdsapi.Client()

request = {
    'product_type': 'reanalysis',
    'variable': 'total_precipitation',
    'year': list(range(2022, 2024)),
    'month': list(range(1, 13)),
    'day': list(range(1, 30)),
    'time': '00:00',
```

```
    'format': 'netcdf',
}


c.retrieve('reanalysis-era5-single-levels', request, 'data/global_precipitation_2022_2023.nc')

import xarray as xr

ds = xr.open_dataset('data/global_precipitation_2022_2023.nc')

ds.to_dataframe()

daily_total_precipitation = ds['tp'].sum(dim=['latitude', 'longitude'])

df = daily_total_precipitation.to_dataframe(name='total_precipitation')

df_reset = df.reset_index()

df_max = df_reset.groupby('time')['total_precipitation'].max().reset_index()
df_max['time'] = pd.to_datetime(df_max['time'])
df_max.sort_values(by='time', inplace=True)

# Interpolate missing dates
date_range = pd.date_range(start=df_max['time'].min(), end=df_max['time'].max(), freq='D')  # 'D' fre
complete_dates = pd.DataFrame(index=date_range)


df_max = complete_dates.merge(df_max, how='left', left_index=True, right_on='time')
df_max['total_precipitation'] = merged_df_max['total_precipitation'].interpolate()
df_max.set_index('time', inplace=True)
df_max.index.name = 'date'

df_max.to_csv('data/merged_df_max_interpolated.csv')

# Extract S&P 500 data
import yfinance as yf
import pandas as pd

gspc = yf.Ticker("^GSPC")
gspc_data = gspc.history(period="max")
print(gspc_data.head())

gspc_data = gspc_data.reset_index()
gspc_subset = gspc_data[['Date', 'Open']].copy()
gspc_subset['Date'] = pd.to_datetime(gspc_subset['Date']).dt.strftime('%Y-%m-%d')
gspc_subset.set_index('Date', inplace=True)
gspc_subset.columns = map(str.lower, gspc_subset.columns)
```

```
print(gspc_subset.head())

gspc_data['Date'] = pd.to_datetime(gspc_data['Date'])
gspc_data.sort_values(by='Date', inplace=True)
date_range = pd.date_range(start=gspc_data['Date'].min(), end=gspc_data['Date'].max(), freq='D')
complete_dates = pd.DataFrame(index=date_range)
merged_data = complete_dates.merge(gspc_data, how='left', left_index=True, right_on='Date')
merged_data['Open'] = merged_data['Open'].interpolate()
merged_data.set_index('Date', inplace=True)
print(merged_data.head())

merged_data.columns = map(str.lower, merged_data.columns)
merged_data_subset = merged_data[['open']].copy()
merged_data_subset.index = pd.to_datetime(merged_data_subset.index).strftime('%Y-%m-%d')
merged_data_subset.index.name = 'date'
print(merged_data_subset.head())

filtered_data = merged_data_subset.loc['2022-01-01':'2023-12-30']
filtered_data.rename(columns={'open': 'sp_500'}, inplace=True)
print(filtered_data.head())
filtered_data.to_csv('data/gspc_open_prices_filtered.csv')

# Daily - weekly comparisons
df_daily = pd.read_csv("data/master_df_2023_12_29.csv").fillna(0)
df_daily["date"] = pd.to_datetime(df_daily["date"])
df_daily = df_daily.set_index("date")
y_daily = df_daily["web_billings"]
X_daily = df_daily.filter(regex = "spend", axis = 1).drop(columns=["mobile_adwords_spend",
                                                                   "asa_spend",
                                                                   "android_mobile_affiliates_spend",
                                                                   "ios_mobile_affiliates_spend"])
X_daily

temperature = pd.read_csv("data/world_avg_temperature.csv").fillna(13.457)[:-3]
temperature["date"] = pd.to_datetime(temperature["date"])
temperature = temperature.set_index("date")

sp_500 = pd.read_csv("data/gspc_open_prices_filtered.csv")
sp_500["date"] = pd.to_datetime(sp_500["date"])
sp_500 = sp_500.set_index("date")

precipitation = pd.read_csv("data/merged_df_max_interpolated.csv")
precipitation["date"] = pd.to_datetime(precipitation["date"])
precipitation = precipitation.set_index("date")

X_daily = pd.merge(X_daily, temperature, left_index=True, right_index=True)
X_daily = pd.merge(X_daily, precipitation, left_index=True, right_index=True)
```

```
X_daily = pd.merge(X_daily, sp_500, left_index=True, right_index=True)
X_daily.tail()

dates = pd.to_datetime(X_daily.reset_index().set_index("date", drop = False)['date'])
dates.dt.day_of_week
X_daily['is_weekend'] = ((dates.dt.dayofweek >= 5) & (dates.dt.dayofweek != 4)).astype(int)

X_daily

split_date = int(len(X_daily) * 0.8)

# Train set
X_train = X_daily.iloc[:split_date + 1]
y_train = y_daily.iloc[:split_date + 1]

# Test set
X_test = X_daily.iloc[split_date + 1:]
y_test = y_daily.iloc[split_date + 1:]

X_daily = X_train
y_daily = y_train

X_weekly_sums = X_daily.drop(columns=["is_weekend", "temperature", "sp_500", "total_precipitation"]).

X_weekly_avgs = X_daily[["temperature", "sp_500", "total_precipitation"]].resample('W-Mon').mean()

X_weekly = pd.merge(X_weekly_sums, X_weekly_avgs, left_index=True, right_index=True)
X_weekly

y_weekly = y_daily.resample('W-Mon').sum()
y_weekly

X_daily.describe()
X_weekly.describe()

y_daily.describe()
y_weekly.describe()

daily_std = X_daily.std()
weekly_std = X_weekly.std()

daily_mean = X_daily.mean()
weekly_mean = X_weekly.mean()

# Coefficient of Variation
cv_daily = (daily_std / daily_mean) * 100
cv_weekly = (weekly_std / weekly_mean) * 100
```

```
print(np.round(cv_daily,2), np.round(cv_weekly,2))

import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt

# Assuming 'X_daily', 'X_weekly', 'y_daily', and 'y_weekly' are pandas DataFrames or Series
# Ensure that the datasets are properly prepared with relevant columns

# Descriptive Statistics
daily_stats = X_daily.describe()
weekly_stats = X_weekly.describe()

# Visualizations
plt.figure(figsize=(12, 6))
plt.plot(X_daily.index, y_daily, label='Daily Data', alpha=0.5)
plt.plot(X_weekly.index, y_weekly, label='Weekly Data', linewidth=2)
plt.title('Comparison of Daily and Weekly Data')
plt.xlabel('Time')
plt.ylabel('Target Variable')
plt.legend()
plt.show()

# CV
cv_daily = (X_daily.std() / X_daily.mean()) * 100
cv_weekly = (X_weekly.std() / X_weekly.mean()) * 100

# OLS quick check
X_daily = sm.add_constant(X_daily)
X_weekly = sm.add_constant(X_weekly)
daily_model = sm.OLS(y_daily, X_daily).fit()
weekly_model = sm.OLS(y_weekly, X_weekly).fit()

# Residuals
daily_residuals = daily_model.resid
weekly_residuals = weekly_model.resid

# Plot Residuals
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(X_daily.index, daily_residuals, label='Daily Residuals', alpha=0.5)
plt.title('Residuals for Daily Data')
plt.xlabel('Time')
plt.ylabel('Residuals')
plt.legend()
```

```
plt.subplot(2, 1, 2)
plt.plot(X_weekly.index, weekly_residuals, label='Weekly Residuals', alpha=0.5)
plt.title('Residuals for Weekly Data')
plt.xlabel('Time')
plt.ylabel('Residuals')
plt.legend()

plt.tight_layout()
plt.show()

daily_model.summary()
weekly_model.summary()

mean_absolute_percentage_error(y_daily, daily_model.predict(X_daily))
mean_absolute_percentage_error(y_weekly, weekly_model.predict(X_weekly))

def calculate_vif(X):
    vif_data = pd.DataFrame()
    vif_data["Variable"] = X.columns
    vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
    return vif_data

vif_daily = calculate_vif(sm.add_constant(X_train))
print("VIF for Daily Data:")
print(np.round(vif_daily, 2))

vif_weekly = calculate_vif(sm.add_constant(X_train))
print("\nVIF for Weekly Data:")
print(np.round(vif_weekly,2))

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'X_daily' and 'X_weekly' are your matrices of explanatory variables

X_daily = X_daily.rename(columns={
    "affiliates_spend": "Affiliates",
    "influencers_spend": "Influencers",
    "brand_search_adwords_spend": "Branded Search Ads: Google",
    "discovery_adwords_spend": "Google Discovery",
    "display_adwords_spend": "Google Display",
    "nonbrand_search_adwords_spend": "Nonbrand Search Ads: Google",
    "youtube_adwords_spend": "YouTube Ads",
    "brand_bing_spend": "Branded Search Ads: Bing",
    "nonbrand_bing_spend": "Nonbrand Search Ads: Bing",
```

```
    "temperature": "Temperature",
    "sp_500": "S&P 500",
    "total_precipitation": "Precipitation",
    "is_weekend": "Weekend dummy"
})

X_weekly = X_weekly.rename(columns={
    "affiliates_spend": "Affiliates",
    "influencers_spend": "Influencers",
    "brand_search_adwords_spend": "Branded Search Ads: Google",
    "discovery_adwords_spend": "Google Discovery",
    "display_adwords_spend": "Google Display",
    "nonbrand_search_adwords_spend": "Nonbrand Search Ads: Google",
    "youtube_adwords_spend": "YouTube Ads",
    "brand_bing_spend": "Branded Search Ads: Bing",
    "nonbrand_bing_spend": "Nonbrand Search Ads: Bing",
    "temperature": "Temperature",
    "sp_500": "S&P 500",
    "total_precipitation": "Precipitation"
})


# Calculate correlation matrices
corr_daily = X_daily.corr()
corr_weekly = X_weekly.corr()

# Plot correlogram for daily data
plt.figure(figsize=(12, 8))
sns.heatmap(corr_daily, cmap='coolwarm', annot=True, fmt=".2f", linewidths=.5)
plt.title('Correlogram for Explanatory Variables - Daily Data')
plt.show()

# Plot correlogram for weekly data
plt.figure(figsize=(12, 8))
sns.heatmap(corr_weekly, cmap='coolwarm', annot=True, fmt=".2f", linewidths=.5)
plt.title('Correlogram for Explanatory Variables - Weekly Data')
plt.show()

# OLS

import os

cwd = os.getcwd()
root_folder = os.path.dirname(cwd)
os.chdir(root_folder)

import matplotlib.pyplot as plt
```

```
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor

def combine_columns(column_1: str, column_2: str, df: pd.DataFrame):
    df_new = df.copy()

    df_new[column_1 + "_" + column_2] = df_new[column_1] + df_new[column_2]
    df_new = df_new.drop(columns=[column_1, column_2])

    return df_new

def calculate_vif(X):
    vif_data = pd.DataFrame()
    vif_data["feature"] = X.columns

    # calculating VIF for each feature
    vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]

    return vif_data

def vif_and_correlation(X):

    vif_daily = calculate_vif(X)
    print("VIF for Daily Data (without constant term):")
    print(vif_daily)


    corr_daily = X.corr()

    plt.figure(figsize=(12, 8))
    sns.heatmap(corr_daily, cmap='coolwarm', annot=True, fmt=".2f", linewidths=.5)
    plt.title('Correlogram for Explanatory Variables - Daily Data')
    plt.show()

y_daily = pd.read_csv("data/master_df_2023_12_29.csv", index_col="date").fillna(0)["web_billings"]


temperature = pd.read_csv("data/world_avg_temperature.csv", index_col="date").fillna(13.457)[:-3]
sp_500 = pd.read_csv("data/gspc_open_prices_filtered.csv", index_col="date")
precipitation = pd.read_csv("data/merged_df_max_interpolated.csv", index_col="date")
```

```
df_daily = pd.read_csv("data/master_df_2023_12_29.csv", sep=",").fillna(0)
df_daily["date"] = pd.to_datetime(df_daily["date"])
df_daily = df_daily.set_index("date")
y_daily = df_daily["web_billings"]
X_daily = df_daily.filter(regex = "spend", axis = 1).drop(columns=["mobile_adwords_spend",
                                                                   "asa_spend",
                                                                   "android_mobile_affiliates_spend",
                                                                   "ios_mobile_affiliates_spend"])
                                                                #    "brand_search_adwords_spend",
                                                                #    "nonbrand_search_adwords_spend",
                                                                #    "discovery_adwords_spend",
                                                                #    "display_adwords_spend",
                                                                #    "youtube_adwords_spend"])

X_daily = pd.read_csv("data/features_regrouped.csv")
X_daily["date"] = pd.to_datetime(X_daily["date"])
X_daily = X_daily.set_index("date")


temperature = pd.read_csv("data/world_avg_temperature.csv").fillna(13.457)[:-3]
temperature["date"] = pd.to_datetime(temperature["date"])
temperature = temperature.set_index("date")

sp_500 = pd.read_csv("data/gspc_open_prices_filtered.csv")
sp_500["date"] = pd.to_datetime(sp_500["date"])
sp_500 = sp_500.set_index("date")

precipitation = pd.read_csv("data/merged_df_max_interpolated.csv")
precipitation["date"] = pd.to_datetime(precipitation["date"])
precipitation = precipitation.set_index("date")

X_daily = pd.merge(X_daily, temperature, left_index=True, right_index=True)
X_daily = pd.merge(X_daily, precipitation, left_index=True, right_index=True)
X_daily = pd.merge(X_daily, sp_500, left_index=True, right_index=True)

dates = pd.to_datetime(X_daily.reset_index().set_index("date", drop = False)['date'])
dates.dt.day_of_week
X_daily['is_weekend'] = ((dates.dt.dayofweek >= 5) & (dates.dt.dayofweek != 4)).astype(int)


# X_daily = X_daily.rename(columns={
#     "affiliates_spend": "Affiliates",
#     "influencers_spend": "Influencers",
#     "brand_search_adwords_spend": "Branded Search Ads: Google",
```

```
#     "discovery_adwords_spend": "Google Discovery",
#     "display_adwords_spend": "Google Display",
#     "nonbrand_search_adwords_spend": "Nonbrand Search Ads: Google",
#     "youtube_adwords_spend": "YouTube Ads",
#     "brand_bing_spend": "Branded Search Ads: Bing",
#     "nonbrand_bing_spend": "Nonbrand Search Ads: Bing",
#     "temperature": "Temperature",
#     "sp_500": "S&P 500",
#     "total_precipitation": "Precipitation",
#     "is_weekend": "Weekend dummy"
# })

X_daily = X_daily.rename(columns={
   "affiliates": "Affiliates",
    "influencers": "Influencers",
    "banners_and_youtube": "Banners and YouTube",
    "paid_search": "Paid Search",
    "temperature": "Temperature",
    "sp_500": "S&P 500",
    "total_precipitation": "Precipitation",
    "is_weekend": "Weekend dummy"
})

def min_max_normalize(df):
    df_normalized = (df - df.min()) / (df.max() - df.min())
    return df_normalized


X_unscaled = X_daily
y_unscaled = y_daily

# X_daily = min_max_normalize(X_daily)
# y_daily = min_max_normalize(y_daily)

keys = {}

for i in X_daily.columns:
    keys[i] = np.random.rand()

for column in X_daily.columns:
    X_daily[column] = X_daily[column] * keys[column]

file_path = 'data/keys_final.json'

# Save the dictionary to a JSON file
with open(file_path, 'w') as json_file:
    json.dump(keys, json_file)
```

```
split_date = int(len(X_daily) * 0.8)

# Train set
X_train = X_daily.iloc[:split_date + 1]
y_train = y_daily.iloc[:split_date + 1]

# Test set
X_test = X_daily.iloc[split_date + 1:]
y_test = y_daily.iloc[split_date + 1:]

X_train = X_train.drop(columns = ["Weekend dummy", "Precipitation"])

daily_model = sm.OLS(y_train, sm.add_constant(X_train)).fit()

daily_model.summary()

X_train = combine_columns("brand_bing", "google_search_brand", X_train)
X_train = combine_columns("nonbrand_bing", "google_search_nonbrand", X_train)

daily_model = sm.OLS(y_train, X_train).fit()

daily_model.summary()

vif_and_correlation(X_train)

# X_train = combine_columns("google_display", "google_discovery", X_train)

daily_model = sm.OLS(y_train, X_train).fit()

daily_model.summary()

vif_and_correlation(X_train)

...

### Bayesian modeling

import os

cwd = os.getcwd()
root_folder = os.path.dirname(cwd)
os.chdir(root_folder)

# PyMC
import numpy as np
import pandas as pd
```

```
import pymc as pm
import arviz as az
import pytensor.tensor as pt
import pytensor

import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

# y_daily = pd.read_csv("data/master_df_2023_12_29.csv", index_col="date").fillna(0)["web_billings"]


temperature = pd.read_csv("data/world_avg_temperature.csv", index_col="date").fillna(13.457)[:-3]
sp_500 = pd.read_csv("data/gspc_open_prices_filtered.csv", index_col="date")
precipitation = pd.read_csv("data/merged_df_max_interpolated.csv", index_col="date")




df_daily = pd.read_csv("data/master_df_2023_12_29.csv", sep=",").fillna(0)
df_daily["date"] = pd.to_datetime(df_daily["date"])
df_daily = df_daily.set_index("date")
y_daily = df_daily["web_billings"]



X_daily = pd.read_csv("data/features_regrouped.csv")
X_daily["date"] = pd.to_datetime(X_daily["date"])
X_daily = X_daily.set_index("date")

temperature = pd.read_csv("data/world_avg_temperature.csv").fillna(13.457)[:-3]
temperature["date"] = pd.to_datetime(temperature["date"])
temperature = temperature.set_index("date")

sp_500 = pd.read_csv("data/gspc_open_prices_filtered.csv")
sp_500["date"] = pd.to_datetime(sp_500["date"])
sp_500 = sp_500.set_index("date")

precipitation = pd.read_csv("data/merged_df_max_interpolated.csv")
precipitation["date"] = pd.to_datetime(precipitation["date"])
precipitation = precipitation.set_index("date")


X_daily = pd.merge(X_daily, temperature, left_index=True, right_index=True)
X_daily = pd.merge(X_daily, precipitation, left_index=True, right_index=True)
X_daily = pd.merge(X_daily, sp_500, left_index=True, right_index=True)
```

```
dates = pd.to_datetime(X_daily.reset_index().set_index("date", drop = False)['date'])
dates.dt.day_of_week
X_daily['is_weekend'] = ((dates.dt.dayofweek >= 5) & (dates.dt.dayofweek != 4)).astype(int)

def min_max_normalize(df):
    df_normalized = (df - df.min()) / (df.max() - df.min())
    return df_normalized

X_unscaled = X_daily
y_unscaled = y_daily

X_daily = min_max_normalize(X_daily)
y_daily = min_max_normalize(y_daily)

X_daily = X_daily.rename(columns={
    "affiliates": "Affiliates",
    "influencers": "Influencers",
    "banners_and_youtube": "Banners and YouTube",
    "paid_search": "Paid Search",
    "temperature": "Temperature",
    "sp_500": "S&P 500",
    "total_precipitation": "Precipitation",
    "is_weekend": "Weekend dummy"
})

with open(json_file_path, 'r') as file:
    keys = json.load(file)

for column in X_daily.columns:
    X_daily[column] = X_daily[column] * keys[column]

control_columns = ["Temperature", "Precipitation", "S&P 500", "Weekend dummy"]

media = X_daily.drop(columns=control_columns)
control = X_daily[control_columns].drop(columns=["Precipitation", "Weekend dummy"])

split_date = int(len(X_daily) * 0.8)

# Train set
media_train = media.iloc[:split_date + 1]
control_train = control.iloc[:split_date + 1]
y_train = y_daily.iloc[:split_date + 1]

# Test set
media_test = media.iloc[split_date + 1:]
```

```
control_test = control.iloc[split_date + 1:]
y_test = y_daily.iloc[split_date + 1:]

def carryover(x, strength, length=21):
    w = pt.as_tensor_variable(
        [pt.power(strength, i) for i in range(length)]
    )

    x_lags = pt.stack(
        [pt.concatenate([
            pt.zeros(i),
            x[:x.shape[0]-i]
        ]) for i in range(length)]
    )

    return pt.dot(w, x_lags)

def hill_function(x, kappa, S):
    return (1/(1+(x/kappa)**(-S)))

daily_model = sm.OLS(y_train, sm.add_constant(pd.merge(media_train, control_train, left_index=True, r

daily_model.summary()

plt.plot(y_train[350:550].reset_index(drop=True), label="Actual")
plt.plot(daily_model.predict(sm.add_constant(pd.merge(media_train, control_train, left_index=True, ri
plt.legend()
plt.title("Model Fit")

daily_model = sm.OLS(y_train, sm.add_constant(pd.merge(media_train, control_train, left_index=True, r

priors = pd.DataFrame(data = {"coefs": daily_model.params, "sd": daily_model.bse})

priors

# Nonnformative priors linear regression

with pm.Model() as model_linear:

    target = y_train

    channel_contributions = []
    var_contributions = []


    # Priors for the regression coefficients
    beta_0 = pm.Uniform('beta_0', 0, 1)
```

```
    for channel in media_train.columns:
        coef = pm.Uniform(f'beta_{channel}', 0, 2)

        channel_data = media_train[channel].values

        channel_contributions.append(coef * channel_data)

    for var in control_train.columns:
        coef = pm.Uniform(f'gamma{var}', -5, 5)

        var_data = control_train[var].values

        var_contributions.append(coef * var_data)

    # Expected value of the outcome (log_sales)
    mu = beta_0 + sum(channel_contributions) + sum(var_contributions)


    # Likelihood (sampling distribution) of observations
    sigma = pm.HalfNormal('sigma', sigma=0.05)
    observed = pm.Normal('observed', mu=mu, sigma=sigma, observed=target)

    trace_linear = pm.sample(5000, tune=5000, return_inferencedata=True, random_seed=42)

az.plot_trace(trace_linear)
pm.summary(trace_linear)

# Informative beta priors with min 0. Adstock only

media_train = media_train.drop(columns="Banners and YouTube")

with pm.Model() as model_adstock_only:

    target = y_train

    channel_contributions = []
    var_contributions = []


    # Priors for the regression coefficients
    beta_0 = pm.TruncatedNormal('beta_0', mu=priors.loc['const', 'coefs'], sigma=priors.loc['const',

    for channel in media_train.columns:
        coef = pm.TruncatedNormal(f'beta_{channel}', mu=priors.loc[channel, 'coefs'], sigma=priors.lo
        coef_carryover = pm.Uniform(f'carryover_{channel}', 0, 1)
```

```
        channel_data = media_train[channel].values

        channel_contributions.append(coef * carryover(channel_data, coef_carryover, 14))

    for var in control_train.columns:
        coef = pm.Normal(f'gamma_{var}', mu=priors.loc[var, 'coefs'], sigma=priors.loc[var, 'sd'])

        var_data = control_train[var].values

        var_contributions.append(coef * var_data)

    mu = beta_0 + sum(channel_contributions) + sum(var_contributions)


    # Likelihood (sampling distribution) of observations
    sigma = pm.HalfNormal('sigma', sigma=0.05)
    observed = pm.Normal('observed', mu=mu, sigma=sigma, observed=target)


    trace_adstock_only = pm.sample(5000, tune=5000)

az.plot_trace(trace_adstock_only)
pm.summary(trace_adstock_only)

# ROAS calcs

# Influencers - took from trace summary beta and carryover
x = 0
for i in np.arange(0, 365):
    x += 0.387 * 0.866**i

2.888 * (y_unscaled.max() - y_unscaled.min()) / (X_unscaled["influencers"].max() - X_unscaled["influe

# Same process for other channels. Remains scaled via random constant through this.

with model_adstock_only:
    pm.sample_posterior_predictive(trace_adstock_only, extend_inferencedata=True, random_seed=42)

az.plot_ppc(trace_adstock_only, num_pp_samples=100);

az.plot_posterior(
    trace_adstock_only,
    var_names=['beta_', 'carryover_'],
    filter_vars='like'
)

posterior = pm.sample_posterior_predictive(trace_adstock_only, model_adstock_only)
```

```
predictions = posterior['posterior_predictive']['observed'].mean(axis=0).mean(axis=0)

# media_decomp = pd.DataFrame({i:np.array(trace['posterior']["contribution_"+str(i)]).mean(axis=(0,1)

plt.plot(y_train[].reset_index(drop=True), label="Actual")
plt.plot(predictions[], label = "Predicted")
plt.legend()
plt.title("Model Fit")

def calculate_r_squared(y_observed, y_predicted):
    mean_observed = np.mean(y_observed)

    numerator = np.sum((y_observed - y_predicted)**2)
    denominator = np.sum((y_observed - mean_observed)**2)

    r_squared = 1 - (numerator / denominator)

    return r_squared

calculate_r_squared(y_train.reset_index(drop=True), predictions)

def calculate_mape(y_observed, y_predicted):

    mape = np.mean(np.abs((y_observed - y_predicted) / y_observed)) * 100

    return mape

import numpy as np

def undo_min_max_scaling(scaled_values, original_min, original_max):

    scaled_values = np.array(scaled_values)
    unscaled_values = scaled_values * (original_max - original_min) + original_min
    return unscaled_values

calculate_mape(undo_min_max_scaling(y_train.reset_index(drop=True), y_unscaled.min(), y_unscaled.max(
                undo_min_max_scaling(predictions.to_dataframe()["observed"], y_unscaled.min(), y_unsc

# Add saturation

priors_2 = pm.summary(trace_adstock_only)[["mean", "sd"]]

def hill_function(x, kappa, S):
    return 1/(1+(x/kappa)**(-S))

def hill_function(x, kappa, S):
    return 1/(1+pt.pow(x/kappa, -S))
```

```
# Extremely inefficient
def carryover(x, theta):
    x = pt.as_tensor_variable(x)

    def adstock_geometric_recurrence_theano(index,
                                            input_x,
                                            decay_x,
                                            theta):
        return pt.set_subtensor(decay_x[index],
               pt.sum(input_x + theta * decay_x[index - 1]) + theta**2 * decay_x[index - 2] + theta**
                theta**5 * decay_x[index - 5] + theta**6 * decay_x[index - 6])
    len_observed = x.shape[0]
    x_decayed = pt.zeros_like(x)
    x_decayed = pt.set_subtensor(x_decayed[0], x[0])
    x_decayed = pt.set_subtensor(x_decayed[1], x[1] + theta * x_decayed[0])
    x_decayed = pt.set_subtensor(x_decayed[2], x[2] + theta * x_decayed[1] + theta**2 * x_decayed[0])
    x_decayed = pt.set_subtensor(x_decayed[3], x[3] + theta * x_decayed[2] + theta**2 * x_decayed[1]
    x_decayed = pt.set_subtensor(x_decayed[4], x[4] + theta * x_decayed[3] + theta**2 * x_decayed[2]
    x_decayed = pt.set_subtensor(x_decayed[5], x[5] + theta * x_decayed[4] + theta**2 * x_decayed[3]


    output, _ = pytensor.scan(
        fn = adstock_geometric_recurrence_theano,
        sequences = [pt.arange(6, len_observed), x[6:len_observed]],
        outputs_info = x_decayed,
        non_sequences = theta,
        n_steps = len_observed - 6
    )

    return output[-1]

# S - shape
# kappa - half sat point

#  Only thing that allowed it to initialize was setting values in pytensor backend config and even th
with pm.Model() as model:

    pytensor.config.mode = 'FAST_COMPILE'
    pytensor.config.optimizer = 'None'
    target = y_train

    var_contributions = []

    # beta_0 = pm.TruncatedNormal('beta_0', mu=0.003, sigma=0.003, lower=0)
```

```
# coef_affiliates = pm.TruncatedNormal('beta_affiliates', mu=0., sigma=0., lower=0)
# carryover_affiliates = pm.TruncatedNormal('carryover_affiliates', mu=0.082, sigma=0.019, lower=
# kappa_affiliates = pm.Uniform('kappa_affiliates', 0.5, 1)
# S_affiliates = pm.Uniform('S_affiliates', 0.5,3)


# coef_influencers = pm.TruncatedNormal('beta_influencers', mu=0., sigma=0., lower=0)
# carryover_influencers = pm.TruncatedNormal('carryover_influencers', mu=0.082, sigma=0.019, lowe
# kappa_influencers = pm.Uniform('kappa_influencers', 0.5, 1)
# S_influencers = pm.Uniform('S_influencers', 0.5,3)

# coef_google_search_nonbrand = pm.TruncatedNormal('beta_google_search_nonbrand', mu=0., sigma=0.
# carryover_google_search_nonbrand = pm.TruncatedNormal('carryover_google_search_nonbrand', mu=0.
# kappa_google_search_nonbrand = pm.Uniform('kappa_google_search_nonbrand', 0.3, 1)
# S_google_search_nonbrand = pm.Uniform('S_google_search_nonbrand', 0.5,3)

# coef_youtube = pm.TruncatedNormal('beta_youtube', mu=0., sigma=0.015, lower=0)
# carryover_youtube = pm.TruncatedNormal('carryover_youtube', mu=0.519, sigma=0.207, lower=0)
# kappa_youtube = pm.Uniform('kappa_youtube', 0.5, 1)
# S_youtube = pm.Uniform('S_youtube', 0.5,3)

coef_affiliates = pm.TruncatedNormal('beta_affiliates', mu=0., sigma=0., lower=0)
carryover_affiliates = pm.TruncatedNormal('carryover_affiliates', mu=0.077, sigma=0.019, lower=0)
# kappa_affiliates = pm.Uniform('kappa_affiliates', 0.3, 1)
# S_affiliates = pm.Uniform('S_affiliates', 0.5,3)


coef_influencers = pm.TruncatedNormal('beta_influencers', mu=0., sigma=0., lower=0)
carryover_influencers = pm.TruncatedNormal('carryover_influencers', mu=0.869, sigma=0.034, lower=
kappa_influencers = pm.Uniform('kappa_influencers', 0.5, 1)
S_influencers = pm.Uniform('S_influencers', 0.5,3)

coef_paid_search = pm.TruncatedNormal('beta_paid_search', mu=0., sigma=0., lower=0)
# carryover_google_search_nonbrand = pm.TruncatedNormal('carryover_google_search_nonbrand', mu=0.
kappa_paid_search = pm.Uniform('kappa_paid_search', 0.3, 1)
S_paid_search = pm.Uniform('S_paid_search', 0.5,3)

coef_youtube = pm.TruncatedNormal('beta_banners_and_youtube', mu=0., sigma=0., lower=0)
carryover_youtube = pm.TruncatedNormal('carryover_banners_and_youtube', mu=0.494, sigma=0.267, lo
kappa_youtube = pm.Uniform('kappa_banners_and_youtube', 0.3, 1)
S_youtube = pm.Uniform('S_banners_and_youtube', 0.5,3)

# coef_google_display = pm.TruncatedNormal('beta_google_display', mu=0, sigma=0.12, lower=0)
# carryover_google_display = pm.TruncatedNormal('carryover_google_display', mu=0.377, sigma=0.261
# kappa_google_display = pm.Uniform('kappa_google_display', 0.5, 1)
# S_google_display = pm.Uniform('S_google_display', 0.5,3)
```

```
    gamma_temperature = pm.Normal('gamma_temperature', mu=-0., sigma=0.)
    gamma_sp_500 = pm.Normal('gamma_sp500', mu=0., sigma=0.)


    mu = gamma_temperature * control_train["temperature"] +  \
         gamma_sp_500 * control_train["temperature"] + \
         coef_youtube * hill_function(carryover(media_train["banners_and_youtube"], carryover_youtube
         coef_influencers * hill_function(carryover(media_train["influencers"], carryover_influencers
         coef_paid_search * hill_function(pt.as_tensor_variable(media_train["paid_search"]), kappa_pa
         coef_affiliates * carryover(media_train["affiliates"], carryover_affiliates)



    # Likelihood (sampling distribution) of observations
    sigma = pm.HalfNormal('sigma', sigma=0.05)
    observed = pm.Normal('observed', mu=mu, sigma=sigma, observed=target)


    trace = pm.sample(2500, tune=2500, return_inferencedata=True)


# Pointwise estimates for carryover specification

# S - shape
# kappa - half sat point

with pm.Model() as model:

    target = y_train

    var_contributions = []


    coef_affiliates = pm.TruncatedNormal('beta_affiliates', mu=8.255, sigma=0.16, lower=0)
    kappa_affiliates = pm.Uniform('kappa_affiliates', 0.3, 1)
    S_affiliates = pm.Uniform('S_affiliates', 0.5,3)


    coef_influencers = pm.TruncatedNormal('beta_influencers', mu=0.153, sigma=0.024, lower=0)
    kappa_influencers = pm.Uniform('kappa_influencers', 0.5, 1)
    S_influencers = pm.Uniform('S_influencers', 0.5,3)

    coef_paid_search = pm.TruncatedNormal('beta_paid_search', mu=0.076, sigma=0.009, lower=0)
    kappa_paid_search = pm.Uniform('kappa_paid_search', 0.3, 1)
    S_paid_search = pm.Uniform('S_paid_search', 0.5,3)

    coef_youtube = pm.TruncatedNormal('beta_banners_and_youtube', mu=0.01, sigma=0.006, lower=0)
```

```
    kappa_youtube = pm.Uniform('kappa_banners_and_youtube', 0.3, 1)
    S_youtube = pm.Uniform('S_banners_and_youtube', 0.5,3)


    gamma_temperature = pm.Normal('gamma_temperature', mu=-0.071, sigma=0.003)
    gamma_sp_500 = pm.Normal('gamma_sp500', mu=0.047, sigma=0.005)


    mu = gamma_temperature * control_train["temperature"] +  \
         gamma_sp_500 * control_train["sp_500"] + \
         coef_youtube * hill_function(carryover(media_train["banners_and_youtube"], 0.3), kappa_youtu
         coef_influencers * hill_function(carryover(media_train["influencers"], 0.873), kappa_influen
         coef_paid_search * hill_function(pt.as_tensor_variable(media_train["paid_search"]), kappa_pa
         coef_affiliates * hill_function(carryover(media_train["affiliates"], 0.077),



    # Likelihood (sampling distribution) of observations
    sigma = pm.HalfNormal('sigma', sigma=0.05)
    observed = pm.Normal('observed', mu=mu, sigma=sigma, observed=target)


    trace = pm.sample(5000, tune=5000, return_inferencedata=True)

az.plot_trace(trace, divergences=False)
pm.summary(trace)

idata=trace

with model:
    pm.sample_posterior_predictive(idata, extend_inferencedata=True, random_seed=42)


az.plot_ppc(idata, num_pp_samples=100);

az.plot_posterior(
    trace,
    var_names=['beta_', 'carryover_', "S_", "kappa_"],
    filter_vars='like'
)

posterior = pm.sample_posterior_predictive(trace, model)
predictions = posterior['posterior_predictive']['observed'].mean(axis=0).mean(axis=0)

# media_decomp = pd.DataFrame({i:np.array(trace['posterior']["contribution_"+str(i)]).mean(axis=(0,1)

plt.plot(y_train[:].reset_index(drop=True), label="Actual")
```

```
plt.plot(predictions[:], label = "Predicted")
plt.legend()
plt.title("Model Fit")

calculate_mape(undo_min_max_scaling(y_train.reset_index(drop=True), y_unscaled.min(), y_unscaled.max(
```