sales = f(marketing spend, saturation ,adstock) + seas + trend

-> optuna(saturation, adstock)

adstock
best_score = 0

best adstock -> iterate through saturations
for loop (change saturation parameter for each channel at a time) -> get output -> check correlation with target variable -> if correlation > best score, best score

lag
shift the series (pd.Series.shift)

Douglas (UK)
Bayesian MMM for SeaWorld

1) Decompose into seasonal, trend, residuals using Prophet
trend + residuals -> PyMC

2) Market spend -> positive only
seasonality -> positive/negative
3) Priors

DTF

log(Sales) = c1 * log(Price) + seasonality + trend + sigma(ci * similarity)

assumptions: category shares seasonality, so dividing by total removes seasonality
sales of sku i / total sales of all skus = ci * (similarity_j * % share of sku_j * count of all available j) for all j
% of shares = ci * TAE (total assortment effect) for all attributes combined

sku 1 1000 sales
sku 2 50 sales
sku 3 10 sales

share is store independent
store 1 (urban center) -> 10000 units of sku1 ~ 40%
store 2 (rural) -> 10 units of sku1 ~ 40%

make it assortment size independent
if store 1 has an assortment size of 100 -> share scores ~0.01 if all else is even
if store 2 has an assortment size of 10 -> share scores ~0.1 if all else is even

% of shares in week 2, store / % of shares in week1 : LHS = y
% of shares in week 3 / % of shares in week2 : LHS = y

→ [(1 + TAE week2) / (1 + TAE week1)]**alpha, where alpha is assortment elasticity -> RHS
log-linear

y = alpha * log ((1+ TAE_week2) / (1+TAE_week1)) **change in market share of sku i is proportional to change in TAE which is in turn proportional to change of only attribute similarity between products

next step: once alpha is calculated

calculate addition and removal factors

how much demand is transfered to each sku in the assortment

To get contribution of each attrribute:

1) add separate TAE terms for each attribute to the linear model and compare assortment elasticity

OR

2) once we have the readily available TAEs -> run a separate model that gets feature importance of each similarity score in predicting TAE

y=TAE
x = Similarity Scores / Attribute OHE (one hot encode)