Offer Personalization using Temporal Convolutions and Multi-objective Optimization

Anonymous authors
Paper under double-blind review

Abstract

Lately, personalized marketing has become important for retail/e-retail firms due to significant rise in online shopping and market competition. Increase in online shopping and high market competition has led to an increase in promotional expenditure for online retailers, and hence, rolling out optimal offers has become imperative to maintain balance between number of transactions and profit. In this paper, we propose our approach to solve the offer optimization problem at the intersection of consumer, item and time in e-retail setting. To optimize offer, we first build a generalized non-linear model using Temporal Convolutional Network to predict the item purchase probability at consumer level for the given time period. Secondly, we establish the functional relationship between historical offer values and purchase probabilities obtained from the model, which is then used to estimate offer elasticity of purchase probability at consumer item granularity. Finally, we optimize offer values using constraint based multi-objective optimization technique. This paper describes our detailed methodology and presents the results of modelling and optimization across categories.

1 Introduction

In most retail/e-retail settings, promotions play an important role in boosting number of transactions and profit. Also, promotions are used on a daily basis in most of the retail environments including online retailers, supermarkets, fashion retailers, etc. A typical retail firm sells thousands of items and needs to design offer for all items at a given time. The depth of offer design is of very high importance as optimized offer roll out can significantly enhance the business' bottom line. In today's economy, retailers offer hundreds or even thousands promotions simultaneously. Promotions aim to increase sales and traffic, enhance awareness when introducing new items, clear leftover inventory, bolster customer loyalty, and improve competitiveness.

Surprisingly, many retailers still employ a manual process based on intuition and past experience to decide the depth and timing of promotions. The unprecedented volume of data that is now available to retailers presents an opportunity to develop decision support tools that can help retailers improve promotion decisions. The promotion planning process typically involves a large number of decision variables, and needs to ensure that the relevant business constraints (called promotion business rules) are satisfied.

2 Related Work

3 Methodology

We treat each relevant consumer-item as an individual object and shape them into weekly time series data based on historical transactions. In this setup, target value at each time step (week) takes a binary input, 1/0 (purchased/non purchased). *Relevancy* of the consumer-item is defined by items transacted by consumer during training time window. *Positive samples* (purchased/1) are weeks where consumer did transact for an item, whereas *Negative samples* (non purchased/0) are the weeks where the consumer did not buy that item. We apply sliding windows testing routine for generating out of time results. The time series is split into 3 parts - train, validation and test as shown in Table **??**. All our models are built in a multi-object fashion, which allows the gradient movement to happen across all consumer-item combinations split in batches. This enables cross-learning to happen across consumers/items. We then perform Feature Engineering over data splits to generate modelling features. Below are some of the feature groups we perform our experiments with:

- **Datetime:** We use transactional metrics at various temporal cuts like week, month, etc. Datetime related features capturing seasonality and trend are also generated.
- **Consumer-Item Profile:** We use transactional metrics at different granularities like consumer, item, consumer-item, department and aisle. We also create features like Time since first order, Time since last order, time gap between orders, Reorder rates, Reorder frequency, Streak - user purchased the item in a row, Average position in the cart, Total number of orders.
- **Consumer-Item-Time Profile:** We use transactional metrics at the intersection of consumer, item and time. We generate interactions capturing consumer behaviour towards items at a given time.
- **Lagged Offsets:** We use statistical rolling operations like mean, median, quantiles, variance, kurtosis and skewness over temporal regressors for different lag periods to generate offsets.

The model we need to build, thus, should learn to identify similarly behaving time series across latent parameters, and take into account consumer and item variations in comparing different time series. A row in time series is represented by

$$y_{\text{cit}} = f(i_{\text{t}}, c_{\text{t}}, .., c_{\text{t-n}}, ic_{\text{t}}, .., ic_{\text{t-n}}, d_{\text{t}}, .., d_{\text{t-n}}) \tag{1}$$

where y_{cit} is purchase prediction for consumer 'c' for item 'i' at time 't'. i_{t} denotes attributes of item 'i' like category, department, brand, color, size, etc at time 't'. c_{t} denotes attributes of consumer 'c' like age, sex and transactional attributes at time 't'. ic_{t} denotes transactional attributes of consumer 'c' towards item 'i' at time 't'. d_{t} is derived from datetime to capture trend and seasonality at time 't'. 'n' is the number of time lags.

3.1 MODELLING

3.1.1 FEATURE ENGINEERING

From data classification, Figure **??**, we can see that data was sub-divided 4 groups:

- **Static Categorical:** These are categorical features that do not vary with time. This includes consumer attributes like sex, marital status and location along with different item attributes like category, department and brand.
- **Temporal Categorical:** These are categorical features that vary with time. It includes all the datetime related features like week, month of year, etc.
- **Static Continuous:** These features are static but continuous. This includes certain consumer attributes like age and weight, item attributes like size, and certain derived features like target encoded features.
- **Temporal Continuous:** These are time varying continuous features. All consumer and item related traditional attributes like number of orders, add to cart order, etc. falls under this bucket.

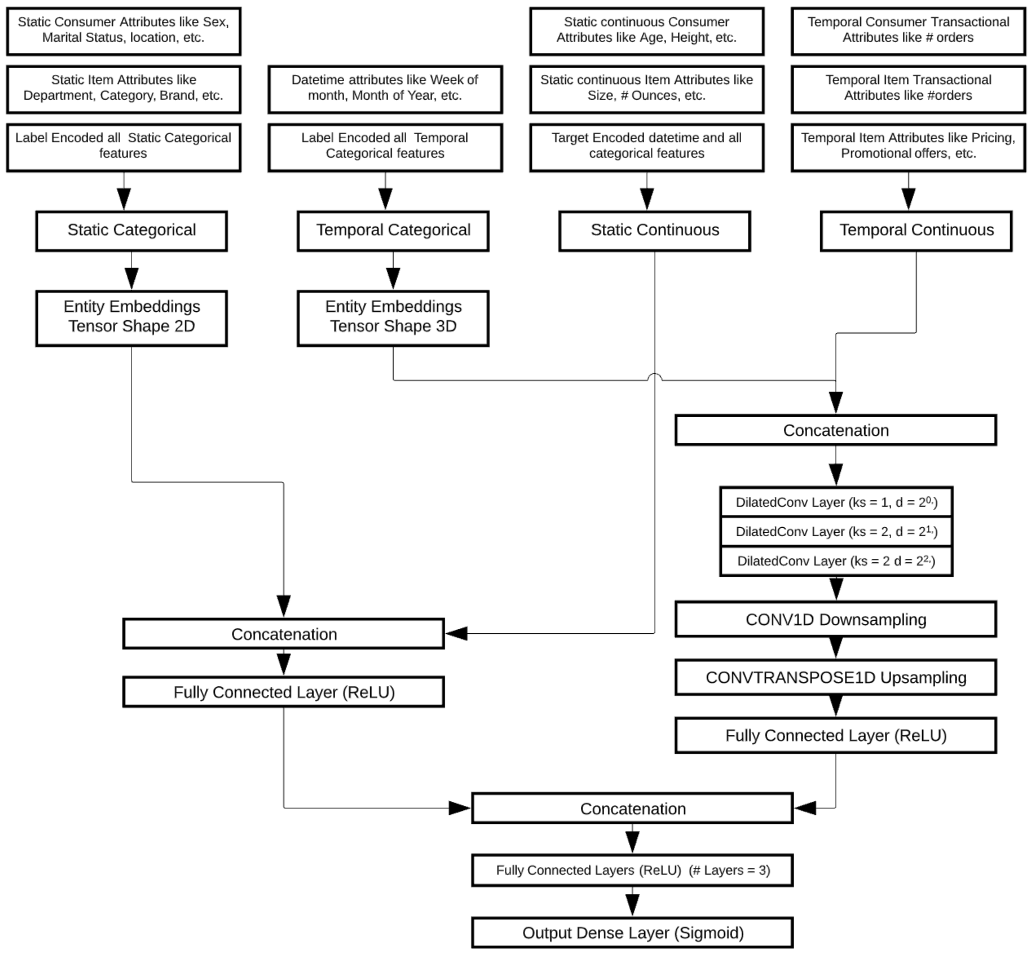
3.1.2 LOSS FUNCTION

Since we are solving Binary Classification problem, we believe that Binary Cross-Entropy should be the most appropriate loss function for training the models. We use the below formula to calculate Binary Cross-Entropy:

$$H_{\text{p}} = -\tfrac{1}{N}\textstyle\sum_{i=1}^{N} y.log(p(y)) + (1-y).log(1-p(y)) \tag{2}$$

here H_{p} represents computed loss, y is the target value (label), and p(y) is the predicted probability against the target. The BCELoss takes non-negative values. We can infer from Equation 2 that Lower the BCELoss, better the Accuracy.

Figure 1: Temporal Convolutional Network (TCN)


Static Consumer Attributes like Sex, Marital Status, location, etc.
Static Item Attributes like Department, Category, Brand, etc.
Label Encoded all Static Categorical features
Datetime attributes like Week of month, Month of Year, etc.
Label Encoded all Temporal Categorical features
Static continuous Consumer Attributes like Age, Height, etc.
Static continuous Item Attributes like Size, # Ounces, etc.
Target Encoded datetime and all categorical features
Temporal Consumer Transactional Attributes like # orders
Temporal Item Transactional Attributes like #orders
Temporal Item Attributes like Pricing, Promotional offers, etc.
Static Categorical
Temporal Categorical
Static Continuous
Temporal Continuous
Entity Embeddings Tensor Shape 2D
Entity Embeddings Tensor Shape 3D
Concatenation
DilatedConv Layer (ks = 1, d = 2^0)
DilatedConv Layer (ks = 2, d = 2^1)
DilatedConv Layer (ks = 2 d = 2^2)
CONV1D Downsampling
CONVTRANSPOSE1D Upsampling
Fully Connected Layer (ReLU)
Concatenation
Fully Connected Layer (ReLU)
Concatenation
Fully Connected Layers (ReLU) (# Layers = 3)
Output Dense Layer (Sigmoid)

3.1.3 Model Architecture

As mentioned earlier in this section, traditional machine learning models are not really a suitable choice for modelling f (Equation 1) due to non-linear interaction between the features. We work with Temporal Convolution Network model as shown in Figure Figure 1.

- **Entity Embeddings + Temporal Convolutional Network:** TCN (Figure 1), originally proposed in **?** , is considered a competitive architecture yielding the best results when evaluated over our experimental dataset. This network comprises of 3 dilated Convolutional networks combined with entity embeddings. Similar to LSTM, this architecture, after convolving and concatenating flows into 3 fully connected ReLU based layers yielding to dense layer which has sigmoid activation.

3.1.4 Hyperparameter Tuning

Hyper-parameters of tree based models are optimized using Bayesian Hyperparameter Optimization Technique, Hyperopt **?**. For Deep learning, we use documented best practices along with our experimental results to choose model hyperparameters. Hyperparameter Optimization is performed over validation dataset. We list some of the hyperparameters along with the values we tune for Deep learning models.

- **Optimizer Parameters:** RMSProp **?** and Adam are used as different trial configurations. The learning rate is experimentally tuned to 1e-3. We also have weight decay of 1e-5 which helps a bit in model Regularization.
- **Scheduler Parameters:** CyclicLR **?** and ReduceLROnPlateau **?** Learning rates are used as different trial configurations. we use 1e-3 as max lr and 1e-6 as base lr for cyclical learning rate along with the step size being the function of length of train loader. ReduceLROn-Plateau is tuned at 1e-6 as min lr.
- **SWA:** Stochastic Weight Averaging (SWA) **?** is used to improve generalization across Deep Learning models. SWA performs an equal average of the weights traversed by SGD with a modified learning rate schedule. We use 1e-3 as SWA learning rate.
- **Parameter Average:** This is a method to average the neural network parameters of n best model checkpoints post training, weighted by validation loss of respective checkpoints. The resulting model generalizes better than those with a single best checkpoint model for an unseen data.

Apart from the above parameters we also iterate to tune network parameters like number of epochs, batch size, number of Fully Connected Layers, number of LSTM layers, convnet parameters (kernel size, dilations, padding) and embedding sizes for the categorical features. Binary Cross-Entropy 2 is used as loss function for all the models.

3.2 F$_1$-MAXIMIZATION

Post stacking, we optimize for purchase probability threshold based on probability distribution at a consumer level using F$_1$-Maximization. This enables optimal thresholding of consumer level probabilities to maximize F$_1$ measure **?**. To illustrate the above, let us say we generated purchase probabilities for 'n' items out of 'b' actually purchased items by consumer 'c'. Now, let us visualize the actuals (3) and predictions (4) of 'n' predicted items for consumer 'c'.

$$A_{\mathrm{c}} = [a_1, a_2, .., a_{\mathrm{n}}] \ \forall \ a_{\mathrm{j}} \in \{0,1\} \tag{3}$$

$$P_{\mathrm{c}} = [p_1, p_2, .., p_{\mathrm{n}}] \ \forall \ p_{\mathrm{j}} \in \ [0,1] \tag{4}$$

A$_c$ represents the actuals for consumer 'c', with a$_j$ being 1/0 (purchased/non purchased). P$_c$ represents the predictions for consumer 'c' for the respective item, with p$_j$ being probability value. 'n' represents total items for which the model generated purchase probabilities for consumer 'c'. Now we apply Decision rule D() which converts probabilities to binary predictions, as described below in Equation 5.

$$D(Pr_{\mathrm{c}}) : {P_{\mathrm{c}}}^{1\,\mathrm{x}\,\mathrm{n}} \rightarrow {P^{'}_{\mathrm{c}}}^{1\,\mathrm{x}\,\mathrm{n}} \quad : p^{'}_{\mathrm{j}} = \begin{cases} 1 & p_{\mathrm{j}} \geq Pr_{\mathrm{c}} \\ 0 & \text{Otherwise} \end{cases} \tag{5}$$

$$P^{'}_{\mathrm{c}} = [p^{'}_1, p^{'}_2, .., p^{'}_{\mathrm{n}}] \ \forall \ p^{'}_{\mathrm{j}} \in \{0,1\} \tag{6}$$

$$k = \textstyle\sum_{i=1}^{n} p^{'}_{\mathrm{i}} \tag{7}$$

Pr$_c$ is the probability cut-off to be optimized for maximizing F$_1$ measure **?** for consumer 'c'. Decision rule D() converts probabilities P$_c$ to binary predictions $P^{'}_c$ such that if p$_j$ is less than Pr$_c$ then $p^{'}_j$ equals 0, otherwise 1. 'k' is the sum of predictions generated post applying Decision rule D(). Now we solve for F$_1$ measure using equations and formulae described below.

$$V_{\mathrm{Pr_c}} = P^{'}_{\mathrm{c}} \ \times \ {A_{\mathrm{c}}}^{\mathrm{T}} \ \Rightarrow \begin{pmatrix} p^{'}_1 & .. & p^{'}_{\mathrm{n}} \end{pmatrix} \times \begin{pmatrix} a_1 \\ .. \\ a_{\mathrm{n}} \end{pmatrix} \tag{8}$$

$$Precision_{\mathrm{c}} = \frac{V_{\mathrm{Pr_c}}}{k} \quad and \quad Recall_{\mathrm{c}} = \frac{V_{\mathrm{Pr_c}}}{b} \tag{9}$$

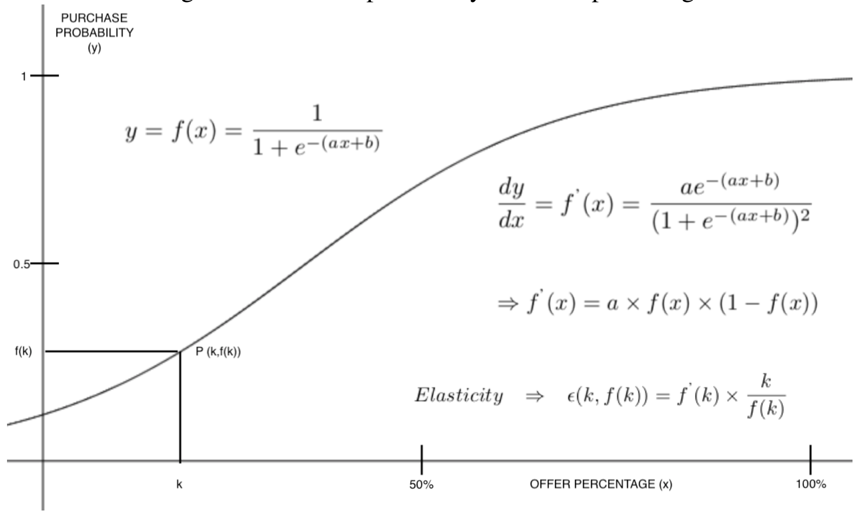
$$F1_{\mathrm{c}} = \frac{2 \times Precision_{\mathrm{c}} \times Recall_{\mathrm{c}}}{Precision_{\mathrm{c}} + Recall_{\mathrm{c}}} \quad \Rightarrow \quad 2 * \frac{V_{\mathrm{Pr_c}}}{k+b} \tag{10}$$

V$_{Pr_c}$ represents the number of items with purchase probabilities greater than Pr$_c$ which were actually purchased (True Positives). As can be seen, Formulae 9 and 10 are used to calculate Precision, Recall and F$_1$-score for consumer 'c'.

$$\max_{V_{\mathrm{Pr_c}}} \quad 2 * \frac{V_{\mathrm{Pr_c}}}{k+b} \quad , \quad \text{subject to:} \quad Pr_{\mathrm{c}} \in \ [0,1] \tag{11}$$

Equation 11 represents the optimization function we solve to generate purchase predictions (1/0) for each consumer.

Figure 2: Purchase probability vs. Offer percentage



3.3 Elasticity Framework

Elasticity estimate is arrived at using simulations. To achieve that, for each consumer-item combination, base offer percent is identified using the following criteria in order:

- Average of last 4 weeks non-zero offer percent values of the consumer-item combination
- Average of historical non-zero offer percent values of the consumer-item combination
- Average of last 4 weeks non-zero offer percent values of all same age consumer-item combinations within that category

$$f(x) = \frac{1}{1+e^{-(ax+b)}} \quad , \quad f^{'}(x) = a \times f(x) \times (1 - f(x)) \tag{12}$$

$$\epsilon(k, f(k)) = f^{'}(k) \times \frac{k}{f(k)} \qquad \Rightarrow \epsilon(k, f(k)) = a \times k \times (1 - f(k)) \tag{13}$$

3.4 Multi-Objective Optimization

The optimization function needs to solve for the below Objectives:

- Maximize the Net Revenue ($\mathrm{R_v}$)
- Maximize the Retention rate ($\mathrm{R_r}$)

$$R_{\mathrm{v}} = \sum_{i=1}^{n} \; [I_{\mathrm{p}} - \frac{I_{\mathrm{p}}}{100} \times (k \pm \eta \times k)] \times \mathbb{1}_{\mathrm{c}}(f(k) \pm \eta \times \epsilon(k, f(k)) \times f(k)) \tag{14}$$

$$R_{\mathrm{r}} = \frac{\sum_{i=1}^{n} \; \mathbb{1}_{\mathrm{c}}(f(k) \pm \eta \times \epsilon(k, f(k)) \times f(k))}{n} \tag{15}$$

$$\mathbb{1}_{\mathrm{c}}(x) := \begin{cases} 1 & x \geq c \\ 0 & \text{Otherwise} \end{cases} \tag{16}$$

Net Revenue ($\mathrm{R_v}$) will be a function of

$$\max_{\eta,\lambda,\gamma} \quad \lambda R_{\mathrm{v}} + \gamma R_{\mathrm{r}} \quad \text{s.t.} \quad : \eta, \lambda, \gamma \in \; (0,1) \quad , \quad \lambda + \gamma = 1 \quad , R_{\mathrm{r}} >= R_{\mathrm{rc}} \tag{17}$$

Figure 3: Category wise Sales Distribution

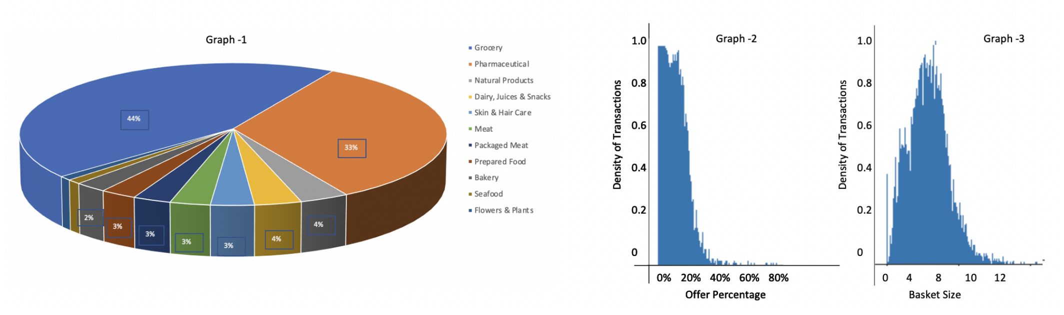


Figure 4: Category graphs

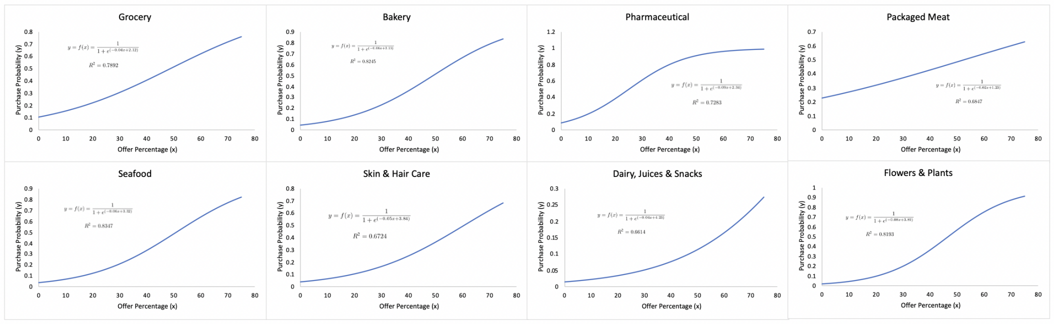


Table 1: Model Results and Elasticity

Category	**Samples**	**BCELoss**	**Precision**	**Recall**	**F_1-Score**	**Avg Elasticity**	**Weighted Offer Percent**
Grocery	28,990	0.0283	0.524	0.501	0.512	1.13	23.37
Bakery	1,628	0.0415	0.346	0.391	0.367	1.06	22.15
Pharmaceutical	20,492	0.0296	0.538	0.483	0.509	0.73	17.89
Packaged Meat	1,473	0.0329	0.473	0.452	0.462	0.62	19.15
Seafood	539	0.0382	0.497	0.379	0.43	0.89	16.23
Natural Products	2,399	0.0319	0.451	0.524	0.485	0.95	21.78
Dairy, Juices, Snacks	2,060	0.0396	0.394	0.492	0.438	1.04	22.02
Prepared Food	1,407	0.0472	0.338	0.295	0.315	1.03	28.27
Skin, Hair Care	1,906	0.0391	0.429	0.453	0.441	0.62	7.93
Meat	1,767	0.0299	0.498	0.524	0.511	0.94	18.92
Flowers, Plants	491	0.0483	0.275	0.318	0.295	1.43	30.24
Vegetables (cut)	124	0.0469	0.364	0.267	0.308	1.53	37.92

4 Experiments and Results

We use transactional data from instacart kaggle challenge to train all our models. As can be seen in Figure **??** data has transactional details including consumer id, item id, order id, add to cart order, date of transaction, aisle id and department id. Also, from Table 1, we can see that we utilize 1 year data which gets split into train, validation, test1 and test2. We generate consumer-item-week level data with purchase/ non purchase being the target. We use the above data to generate consumer-item purchase predictions for 2 time steps in the future (2 weeks in our case).

5 Conclusion

We have presented our study of the consumer purchase behaviour in the context of large scale retail. We have shown that careful feature engineering when used in conjunction with Deep Neural

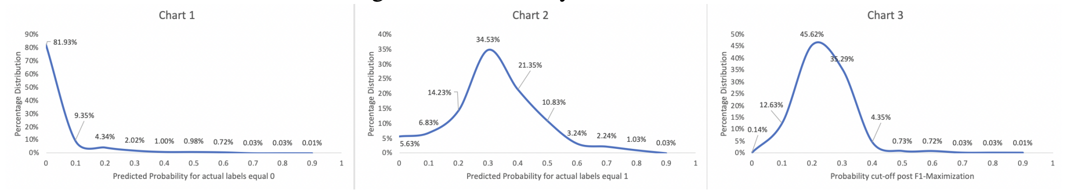


Figure 5: Probability Distributions

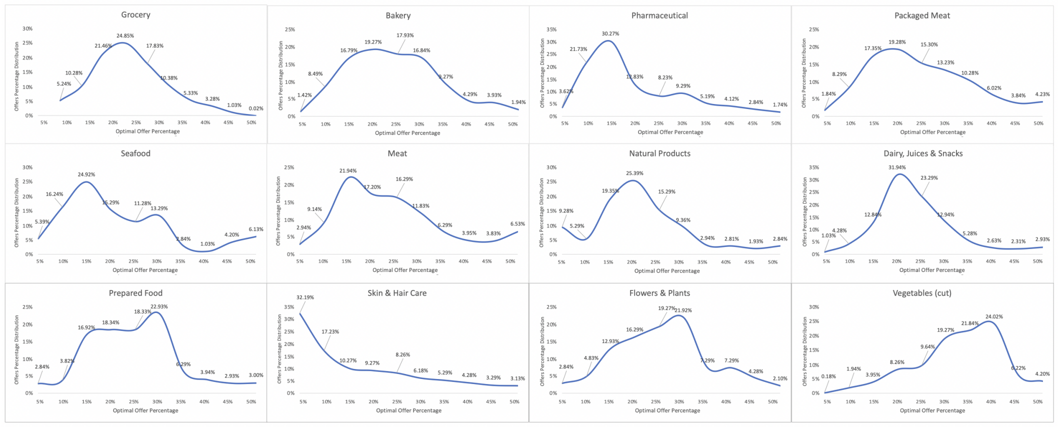


Figure 6: Optimal Offers

Networks, can be used to predict the next (multi-timestep) logical purchase of consumers with reasonably good accuracies. While creating our models and features we have been cognizant of the fact that many features will not be available as is when predictions are being generated for future period. Hence we have used innovative transformations so that we don't have to remember train data during forecast time, thereby reducing the computation and memory requirements during forecast generation.

As per our initial expectations, Deep Neural Network models outperformed the ML models like Xgboost and RandomForest. Sequence to Sequence architectures seemed to be theoretically sound choice for tackling our problem. Our results and observations were inline with the above thought process. Model generalization and robustness was attained by model stacking. As per our expectation we realized gain in accuracy post stacking.

At the same time we understand that computation strategy is key aspect in modelling Millions of customers, and we intend to explore further over this aspect by building Transfer Learning framework **?**. Also, we are working, to further improve our Sequence to Sequence neural network architectures to improve accuracy and decrease computation time.