OFFER PERSONALIZATION USING TEMPORAL CONVOLUTIONS AND MULTI-OBJECTIVE OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Personalized marketing has become important for large scale to small scale e-retail firms due to significant raise 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. Within these firms, the category manager has to solve the promotion optimization problem for each consumer segment and item combination, to be able to design the best offer for each period in a finite horizon, so as to maximize the retailer's profit. This paper considers the problem of planning sales promotions at the intersection of consumer, item and time in e-retail setting, where we first predict item purchase propensity and then estimate the consumer offer elasticity for solving offer optimization. We build deep neural network model to predict purchase propensity, simulation approach for consumer elasticity and finally offer optimization to compute best offer at consumer item level.

1 INTRODUCTION

Consumer behaviour insights have always been one of the key business drivers for retail, given fast changing consumer needs. Existing trend, competitor pricing, item reviews, sales and marketing are some of the key factors driving today's consumer world in retail. While very little information is available on future variablities of the above factors, retailers do have large volumes of historical transactional data. Past study **?** has shown that retailers use conventional techniques with available data to model consumer purchase. While these help in estimating purchase pattern for loyal consumers and high selling items with reasonable accuracy, they do not perform well for the long tail. Since multiple parameters interact non-linearly to define consumer purchase pattern, traditional models are not sufficient to achieve high accuracy across thousands to millions of consumers.

Most retail/e-retail brands, plan their short term inventory (2-4 weeks ahead) based on consumer purchase pattern. Also, certain sales and marketing strategies like Offer Personalization and personalized item recommendations are made leveraging results of consumer purchase predictions for the near future. Given that every demand planner works on a narrow segment of item portfolio, there is a high variability in choices that different planners recommend. Additionally, the demand planners might not get enough opportunities to discuss their views and insights over their recommendations. Hence, subtle effects like cannibalization **?**, and item-affinity remain unaccounted for. Such inefficiencies lead to a gap between consumer needs and item availability, resulting in the loss of business opportunities in terms of consumer churn, and out-of-stock and excess inventory. Our paper makes the following contributions -

- We study and present the usefulness of applying various deep learning architectures along with tree based machine learning algorithms to predict the next logical item purchase at consumer level.
- We present the performance of individual models with varying hyperparameter configurations.
- We implement stacked generalization framework **?** as an ensemble method where a new model learns to combine the predictions from multiple existing models.
- We design and implement F_1-maximization algorithm which optimises for purchase probability cut-off at consumer level.

2 RELATED WORK

In the past few years, usefulness of various machine learning methods for predicting consumer purchase pattern have been analyzed in the academia field and few of them are often used by ML practitioners. In most cases many of those approaches are based on extracting consumer's latent characteristics from its past purchase behavior and applying statistical and ML based formulations **??**. Some previous studies have analyzed the use of random forest and Xgboost techniques to predict consumer retention, where past consumer behavior was used as potential explanatory variable for modelling such patterns. In one such study **?**, the authors develop a model for predicting whether a consumer performs a purchase in prescribed future time frame based on historical purchase information such as the number of transactions, time of the last transaction, and the relative change in total spending of a consumer. They found gradient boosting to perform best over test data. We propose neural network architectures with entity embeddings **?** which outperform the gradient boosting type of models like Xgboost **?**.

From Neural Network architectures perspective, close to our work is Deep Neural Network Ensembles for Time Series Classification **?**. In this paper, authors show how an ensemble of multiple Convolutional Neural Networks can improve upon the state-of-the-art performance of individual neural networks. They use 6 deep learning classifiers including Multi Layer Perceptron, Fully Convolutional Neural Network, Residual Network, Encoder **?**, Multi-Channels Deep Convolutional Neural Networks **?** and Time Convolutional Neural Network **?**. The first three were originally proposed in **?**. We propose the application of such architectures in the consumer choice world and apply the concept of entity embeddings **?** along with neural network architectures like Multi Layer Perceptron, Long Short Term Memory (LSTM), Temporal Convolutional Networks (TCN) **?** and TCN-LSTM **?**.

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 2. 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

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

Table 1: Modelling data splits

Data Split	Specifications	Consumer-Item combinations	Max Time-Series length
Train	Model training	50,872	46 weeks
Validation	HyperParameter Optimization	50,888	2 weeks
Test	Reporting Accuracy Metrics	50,910	2 weeks

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 1, 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:

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

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.

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 2.

- **Entity Embeddings + Temporal Convolutional Network:** TCN (Figure 2), 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.

Figure 1: Data classification for DNN Architectures

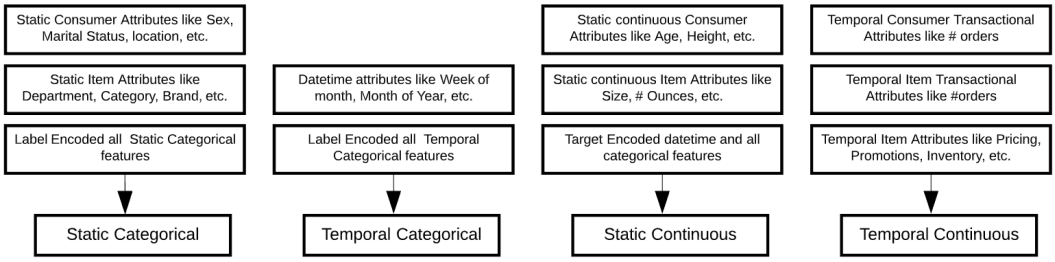
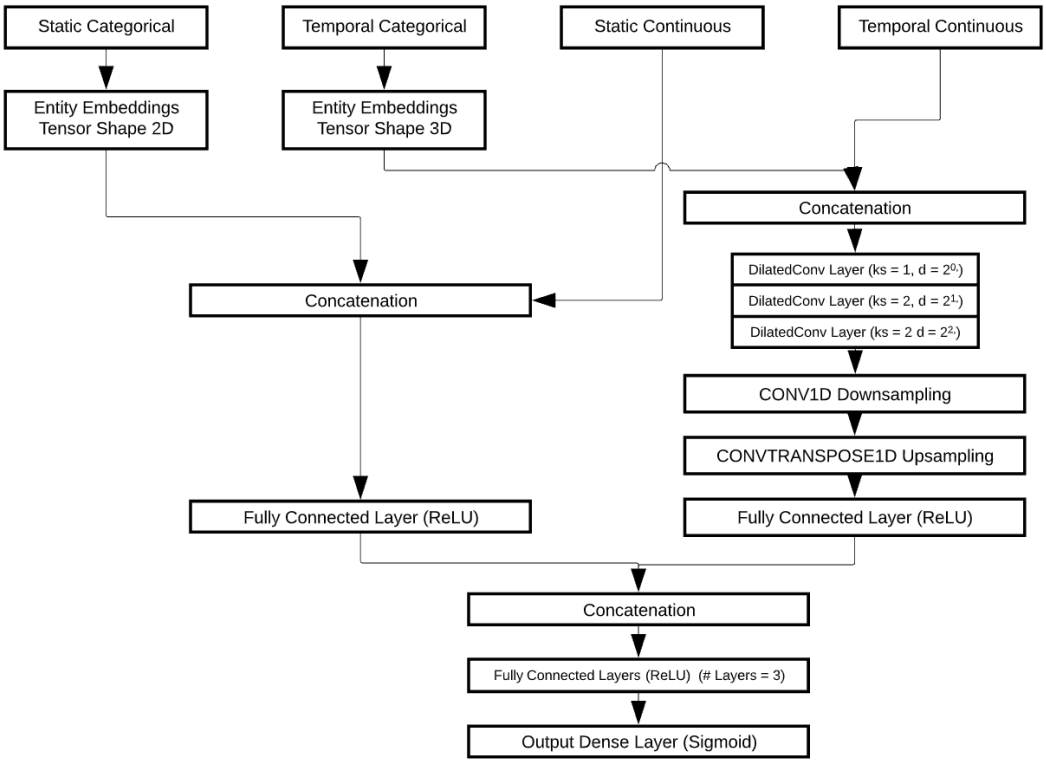


Figure 2: Temporal Convolutional Network (TCN)



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. ReduceLROnPlateau 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'.

$$A_c = [a_1, a_2, .., a_n] \; \forall \; a_j \in \{0,1\} \tag{3}$$

$$P_c = [p_1, p_2, .., p_n] \; \forall \; p_j \in \; [0, 1] \tag{4}$$

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.

$$D(Pr_c) : {P_c}^{1 \times n} \rightarrow {P^{'}}_c^{1 \times n} \quad : {p^{'}}_j = \begin{cases} 1 & p_j \geq Pr_c \\ 0 & \text{Otherwise} \end{cases} \tag{5}$$

$${P^{'}}_c = [{p^{'}}_1, {p^{'}}_2, .., {p^{'}}_n] \; \forall \; {p^{'}}_j \in \{0,1\} \tag{6}$$

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

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.

$$V_{Pr_c} = {P^{'}}_c \; \times \; {A_c}^T \; \Rightarrow \left(\; {p^{'}}_1 \quad .. \quad {p^{'}}_n \; \right) \times \begin{pmatrix} a_1 \\ .. \\ a_n \end{pmatrix} \tag{8}$$

$$Precision_c = \frac{V_{Pr_c}}{k} \quad and \quad Recall_c = \frac{V_{Pr_c}}{b} \tag{9}$$

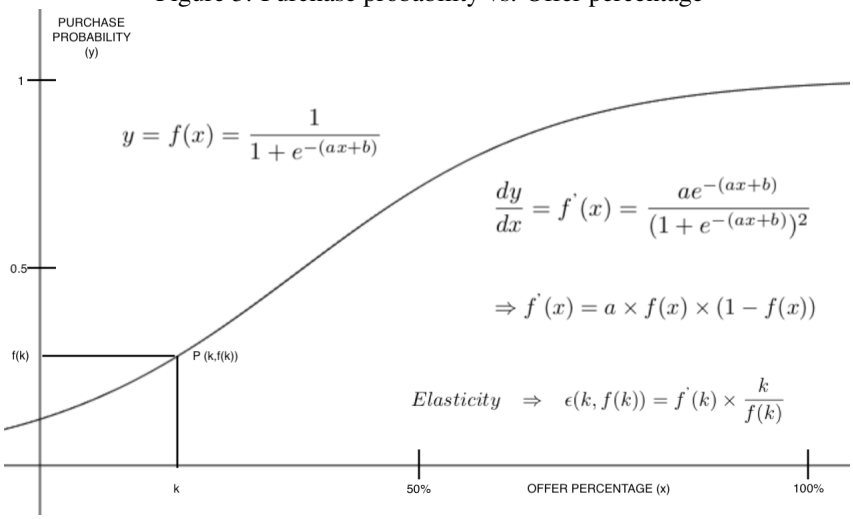
$$F_{1_c} = \frac{2 \times Precision_c \times Recall_c}{Precision_c + Recall_c} \quad \Rightarrow \quad 2 * \frac{V_{Pr_c}}{k+b} \tag{10}$$

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'.

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

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

Figure 3: 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 (R_v)
- Maximize the Retention rate (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 (R_v) will be a function of

Figure 4: Category wise Sales Distribution

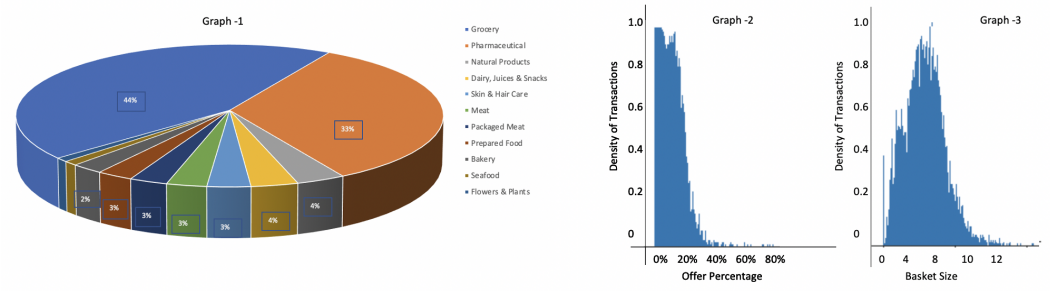
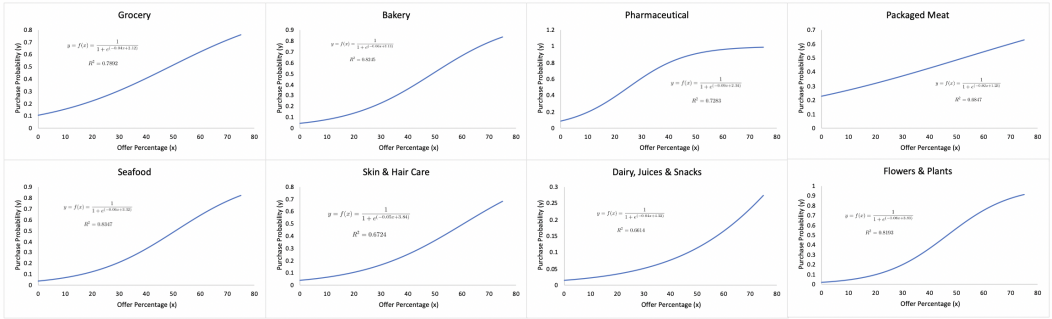


Figure 5: Category graphs



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

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 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.

Figure 6: Probability Distributions

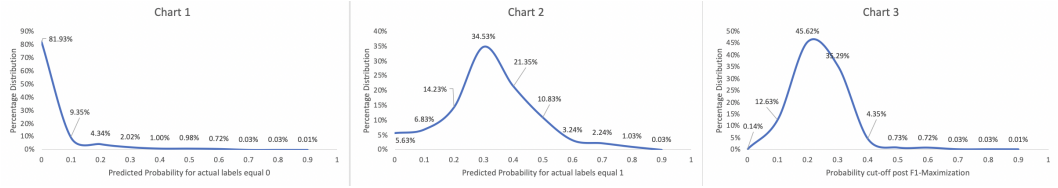


Table 2: Modelling Results

Metric	Train	Validation	Test
BCELoss	0.0298	0.0318	0.0302
Precision	0.512	0.489	0.508
Recall	0.536	0.513	0.523
F_1-Score	0.523	0.501	0.515

Figure 7: Sample Dataset

user_id	order_id	product_id	add_to_cart_order	order_number	date	product_name	aisle_id	department_id	aisle	department
1	2539329	196	1	1	01/01/18	Soda	77	7	soft drinks	beverages
1	2539329	14084	2	1	01/01/18	Organic Unsweetened Vanilla Almond Milk	91	16	soy lactosefree	dairy eggs
1	2539329	12427	3	1	01/01/18	Original Beef Jerky	23	19	popcorn jerky	snacks
1	2539329	26088	4	1	01/01/18	Aged White Cheddar Popcorn	23	19	popcorn jerky	snacks
1	2539329	26405	5	1	01/01/18	XL Pick-A-Size Paper Towel Rolls	54	17	paper goods	household
1	2398795	196	1	2	16/01/18	Soda	77	7	soft drinks	beverages
1	2398795	10258	2	2	16/01/18	Pistachios	117	19	nuts seeds dried fruit	snacks
1	2398795	12427	3	2	16/01/18	Original Beef Jerky	23	19	popcorn jerky	snacks
1	2398795	13176	4	2	16/01/18	Bag of Organic Bananas	24	4	fresh fruits	produce
1	2398795	26088	5	2	16/01/18	Aged White Cheddar Popcorn	23	19	popcorn jerky	snacks
1	2398795	13032	6	2	16/01/18	Cinnamon Toast Crunch	121	14	cereal	breakfast

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.