Consumer Behaviour in Retail: Next Logical Purchase using Deep Neural Network

Ankur Verma
Indian Institute of Technology BHU
ankur.verma.phe09@iitbhu.ac.in

Abstract

Predicting consumer purchase pattern is one of the most challenging problems for large scale retail firms. Retailers spend a lot of money and resources to ensure smooth and delightful shopping experience. This includes recommending relevant items to consumers and maintaining right inventory. The later helps minimise the instances of Out of stock and Excess Inventory. Consumer purchase prediction problem has generally been addressed by ML researchers in conventional manner either through recommender systems or traditional ML approaches. To our knowledge none of the models have generalized well enough in predicting the consumer purchase pattern. In this paper we present our study of consumer purchase behaviour at the intersection of consumer, item and time using e-commerce retail data. For each of the relevant consumer-item combination, we create a sequential time-series data. We then build generalised non-linear model to predict propensity of consumer to purchase the item for given time horizon. We demonstrate robust performance by experimenting with different neural network architectures, ML models and their combination. We showcase the benefits that neural network architectures like Multi Layer Perceptron, Long Short Term Memory and Temporal conventional Networks bring over ML models like Xgboost and RandomForest.

Introduction

Consumer behaviour insights have always been one of the key business drivers for retail, specially given fast changing consumer needs. Existing trend, competitor pricing, item reviews and marketing are some of the key factors driving todays consumer world in retail. While very little information is available on future variablities of the above factors, what retailers have is large volumes of transactional data. Retailers use conventional techniques to model transactional data for predicting consumer choice. While these help in estimating purchase pattern for loyal consumers and high selling items with reasonable accuracy, they don't perform well for the rest. 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.

In many of the retail brands, short term (4-6 weeks ahead) inventory planning is done on the basis of consumer purchase pattern. Given that every demand planner works on a narrow segment of item portfolio, there is high variability in choices that different planners recommend. Also, given their busy schedule, they have very less interaction moments to discuss their views and insights over their recommendations. Hence, subtle effects like item cannibalization, item affinity, pricing remains unaccounted correctly. Such inefficiencies lead to gap between consumer needs and item availablity, resulting in loss of business opportunities in terms of consumer churn, out of stock and excess inventory.

In this paper, we apply multiple deep learning architectures along with tree based machine learning algorithms to predict the next logical purchase at consumer level. We showcase the performance of individual models with varying hyper configurations. We also show the performance of stacked generalization ensemble and F1-maximization which involves combining predictions from different models and fine tuning purchase probability cut-off at consumer level respectively. The following section explains the overall methodology adopted to solve the problem. It also lays out various algorithmic variants and neural network architectures applied to the problem. Finally, section 3 describes the experiments and results obtained in various scenarios of modelling.

Methodology

We treat each consumer-item as an individual object and generate weekly time series based on historical transaction for each object. The target value at each time step (week) takes a binary input, 1/0 (purchased/not purchased). We adopted walk forward Validation strategy for model testing and generalization. We split the data into 4 parts based on the time series as shown in Table 1. We then generate various types of features including datetime related, label encoded and target encoded within and across objects. Below are the feature groups we experimented with explicitly or unexplicitly:

- **Datetime:** Transactional metrics at various temporal cuts including week, month and quarter. Datetime related features capturing seasonality and trend.

Table 1: Modelling data splits

Data Split	Specifications	Consumer-Item combinations	Max Time-Series length
Train	Model training	2,279	52 weeks
Validation	Hyper-Parameter Optimization	2,359	4 weeks
Test1	Stacked Generalization, F_1-Maximization	2,394	4 weeks
Test2	Reporting Accuracy Metrics	2,421	4 weeks

Figure 1: Multi Layer Perceptron (MLP)

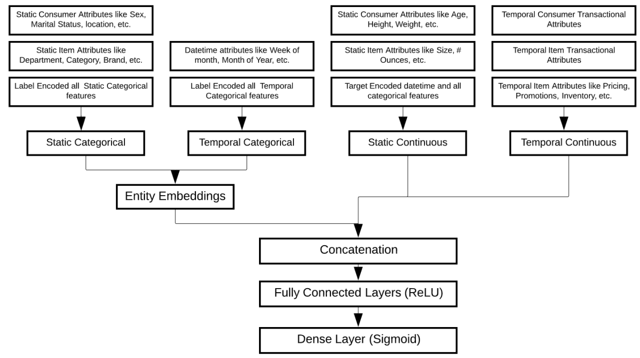
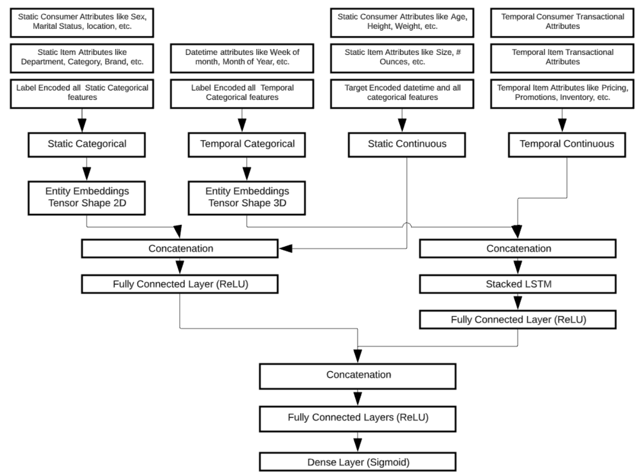


Figure 2: Long Short Term Memory (LSTM)



- **Consumer-Item Profile:** Transactional metrics at different granularities including consumer, item, consumer-item, department, aisle, etc. The metrics includes some of likes 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:** Transactional metrics at the intersection of consumer, item and time. Interactions capturing consumer behaviour towards items for the given time period.

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

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

where y_{cit} is sales for consumer 'c' item 'i' at time 't'. i_t is attribute of the item 'i' like category, department, brand, color, size, etc. c_t is attribute of the consumer 'c' like age, sex and transactional attributes. ic_t is transactional attributes of the consumer 'c' towards item 'i'. d_t is derived from datetime to capture trend and seasonality. Finally, n is the number of time lags.

Loss Function

Since we are solving Binary classification problem, Binary Cross-Entropy/Log Loss seemed most logical loss function for training all our models.

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

where y is the label and p(y) is the predicted probability.

Model Architectures

As mentioned in previous section, traditional machine learning models are not suitable choice for solving Equation 1. Hence, we work with machine learning tree based models like Random Forest Gradient Boosted Trees to Deep learning models ranging from Multi Layer Perceptron (MLP), Long Short Term Memory (LSTM) and Temporal Convolutional Network (TCN). Architectures of MLP, LSTM, TCN and TCN-LSTM models are shown in Figure 1, Figure 2, Figure 3 and Figure **??**.

The consumer purchase pattern has huge variation in terms of time of purchase (weekday/weekends), cadence of purchase (days to months), purchased item types (dairy/meat/grocery/apparels/etc.) and brand loyalty (tendency to substitute items). Given such huge variance it becomes imperative to cross learn consumer behaviour from like consumer groups. To learn such relationships its very important to capture non-linear relationship between target and regressors at the most granular level. Tree based and Deep learning models are chosen for their ability to model feature interactions even if transient in time, so that they capture non-linearity well. Utility of traditional machine learning algorithms like Logistic regression , SVM and recommender systems are limited given the scale of large retail firms (Millions of customers with thousands of products). Such models do not scale well for large sets of data and hyperparameters.

Tree based models and MLP are trained in such a way where lagged values of time varying features are used to capture temporal dependencies. We use lagged values of temporal features up to last n time steps (n goes till 52 weeks). Multiple Lagged values as well as Statistical rolling operations like mean, median, quantiles, variance, kurtosis and skewness over varying lag periods are used for feature generation. Details around datasets and derived features are explained in the following section. This was decided after some preliminary experiments. Hyper-parameters of tree based models are optimized using Bayesian Hyper-

Figure 3: Temporal Convolutional Network (TCN)

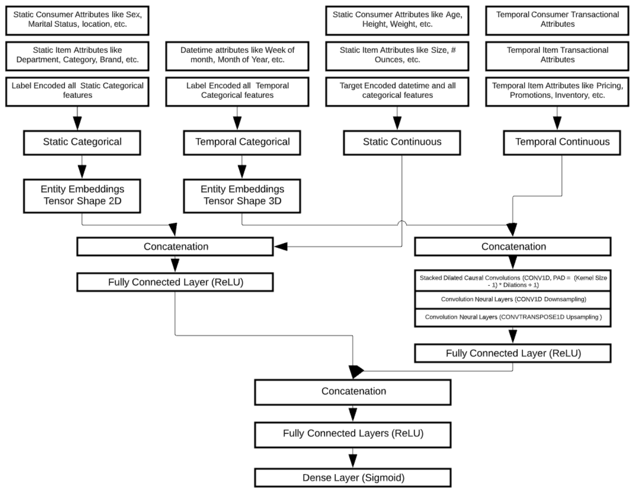
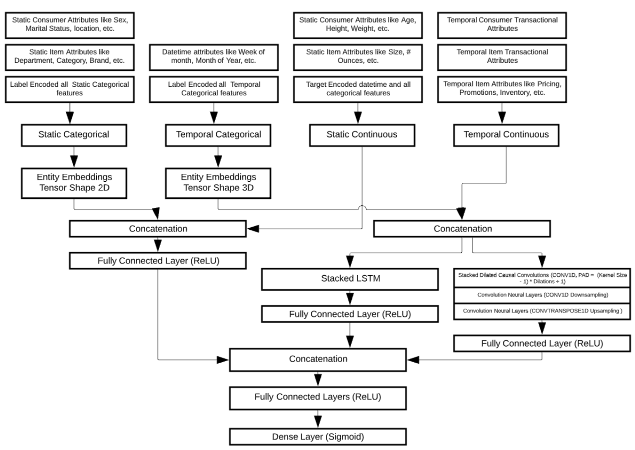


Figure 4: Temporal Convolutional Network + Long Short Term Memory (TCN-LSTM)



parameter Optimization Technique. LSTM, TCN and TCN-LSTM models were trained in sequence to sequence fashion using entire life-cycle data of a time series (Consumer-Item).

We applied empirical approach to tune the hyperparameters for Deep learning models. All hyperparameter Optimization was performed over Validation dataset. We list some of the params along with the values we used for Deep learning models.

- **Optimizer Parameters:** RMSProp and Adam were used as different trial configurations. The learning rate was experimentally tuned to 1e-3. We also did weight decay of 1e-5 which helped a bit in model Regularization.
- **Scheduler Parameters:** Cyclic and ReduceLROnPlateau Learning rates were used as different trial configurations. we used 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 was tuned for 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 used 1e-3 as SWA learning rate.

Apart from the above parameters we also iterated enough 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/Log Loss [2] was used as loss function for all the models. Deep learning models are built using deep learning framework PyTorch, and are trained on GCP instance containing 6 CPUs and a single GPU. scikit-learn is used for Tree based models like RandomForest and Xgboost. We built a total of 60 models, 12 different param configurations for each of 4 Deep Learning models and 6 best trials for each of 2 Machine Learning models as shown in table 1.

Stacked Generalization

For simplicity and better generalization we adopted non-parameteric approach for stacking. We used Weighted K-Best and K-Best as Stacked Generalization Ensemble model for combining the 60 models (candidates) from Table 1. Test1 BCELoss was used as metric to compute weight for each of the 60 candidates. We set k to 5, 10 and 25 and selected these top candidates based upon the experiments performed over test1 metrics. Finally, we took Weighted Average of the probabilities for Train, Validation, Test1 and Test2 time steps for each consumer-item combination.

F1-Maximization

Post generation of the consumer-item probabilities, we optimize for the purchase cut-off probability based on probability distribution of test1 at consumer level. For instance, lets say we generated purchase probabilities for n_i items of b_i actually purchased items for consumer c_i. Let Actual items purchased by consumer c_i at test1 time step be [Ia_1, Ia_2 ,.., Ia_{b_i}] whereas items for which the model generated probabilities for the consumer c_i at test1 time step be [Ip_1, Ip_2 ,.., Ip_{n_i}].

$$A_{c_i} = [a_1, a_2, .., a_{n_i}] \; \forall \; a_j \in \{0,1\} \tag{3}$$

$$P_{c_i} = [p_1, p_2, .., p_{n_i}] \; \forall \; p_j \in \; [0, 1] \tag{4}$$

A_{c_i} represents the actuals for consumer c_i, with a_j being 1/0 (purchased/non purchased). P_{c_i} represents the predicted probabilities for consumer c_i for respective item, with p_j being probability value. As mentioned above n_i is the total items model generated purchase probabilities for.

$$D(Pr_{c_i}) : {P_{c_i}}^{1 \text{ x } n_i} \rightarrow {P'_{c_i}}^{1 \text{ x } n_i} \;\; p'_j = \{1 \text{ if } p_j \geq Pr_{c_i}\} \tag{5}$$

$$P'_{c_i} = [p'_1, p'_2, .., p'_{n_i}] \; \forall \; p'_j \in \{0,1\} \tag{6}$$

Table 2: Model Specifications

Model Type	Trials	Model Hyper-Parameters	Loss Functions
MLP	12	Optimizer, Scheduler, SWA, Parameter Averaging, Feature Groups, FC Layers	BCELoss
LSTM	12	Optimizer, Scheduler, SWA, Parameter Averaging, Feature Groups, FC Layers, LSTM Layers	BCELoss
TCN	12	Optimizer, Scheduler, SWA, Parameter Averaging, Feature Groups, FC Layers, Convolution Parameters	BCELoss
TCN-LSTM	12	Optimizer, Scheduler, SWA, Parameter Averaging, Feature Groups, FC Layers, LSTM, Convolution Parameters	BCELoss
Xgboost	6	Learning rate, Tree Depth, Regularization parameters	Logloss
RandomForest	6	Tree Depth, Evaluation Metrics, Regularization parameters	Entropy, Gini

Figure 5: Sample Dataset

user_id	order_id	order_number	product_id	add_to_cart_order	date	product_name	aisle_id	department_id	aisle	department
1	431534	5	196	1	04/04/18	Soda	77	7	soft drinks	beverages
1	431534	5	12427	2	04/04/18	Original Beef Jerky	23	19	popcorn jerky	snacks
1	431534	5	10258	3	04/04/18	Pistachios	117	19	nuts seeds dried fruit	snacks
1	431534	5	25133	4	04/04/18	Organic String Cheese	21	16	packaged cheese	dairy eggs
1	431534	5	10326	5	04/04/18	Organic Fuji Apples	24	4	fresh fruits	produce
1	431534	5	17122	6	04/04/18	Honeycrisp Apples	24	4	fresh fruits	produce
1	431534	5	41787	7	04/04/18	Bartlett Pears	24	4	fresh fruits	produce
1	431534	5	13176	8	04/04/18	Bag of Organic Bananas	24	4	fresh fruits	produce
1	3367565	6	196	1	23/04/18	Soda	77	7	soft drinks	beverages
1	3367565	6	12427	2	23/04/18	Original Beef Jerky	23	19	popcorn jerky	snacks
1	3367565	6	10258	3	23/04/18	Pistachios	117	19	nuts seeds dried fruit	snacks
1	3367565	6	25133	4	23/04/18	Organic String Cheese	21	16	packaged cheese	dairy eggs
1	550135	7	196	1	13/05/18	Soda	77	7	soft drinks	beverages
1	550135	7	10258	2	13/05/18	Pistachios	117	19	nuts seeds dried fruit	snacks
1	550135	7	12427	3	13/05/18	Original Beef Jerky	23	19	popcorn jerky	snacks
1	550135	7	25133	4	13/05/18	Organic String Cheese	21	16	packaged cheese	dairy eggs
1	550135	7	13032	5	13/05/18	Cinnamon Toast Crunch	121	14	cereal	breakfast

Pr_{c_i} is the probability cut-off. Decision rule D converts probabilities P_{c_i} to binary predictions P'_{c_i} such that if p_j is less than Pr_{c_i} then p'_j equals 0 else 1.

$$V_{Pr_{c_i}} = P'_{c_i} \times A_{c_i}{}^{T} \Rightarrow \begin{pmatrix} p'_1 & .. & p'_{n_i} \end{pmatrix} \times \begin{pmatrix} a_1 \\ .. \\ a_{n_i} \end{pmatrix} \tag{7}$$

$V_{Pr_{c_i}}$ represents the number of items with purchase probabilities greater than Pr_{c_i} which were actually purchased. Now using the below formulae we calculate Precision, Recall and F_1-score for consumer c_i.

$$Precision_{c_i} = \frac{V_{Pr_{c_i}}}{n_i} \quad and \quad Recall_{c_i} = \frac{V_{Pr_{c_i}}}{b_i} \tag{8}$$

$$F_{1_{c_i}} = \frac{2 \times Precision_{c_i} \times Recall_{c_i}}{Precision_{c_i} + Recall_{c_i}} \quad \Rightarrow \quad 2 * \frac{V_{Pr_{c_i}}}{n_i + b_i} \tag{9}$$

$F_{1_{c_i}}$ becomes the value to be maximised by finding optimal Pr_{c_i} for consumer c_i. We apply the above Optimization function over test1 probability distribution at a consumer level to find optimal cut-off. Final purchase predictions happens based on the cut-off value.

Experiments and Results

We use transactional data from instacart kaggle challenge to train all our models. A preview of data used is shown in Figure 5.

Figure 6: Most ordered Items across Departments

Figure 7: Density of consumers Vs. Basket Size

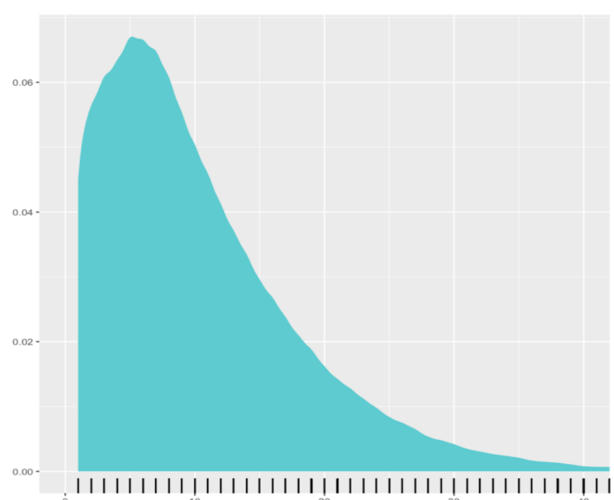


Table 3: BCELoss of Test2 for 12 Trials of Deep Learning Models

Trial	Optimizer	Scheduler	SWA	Parameter Avg	MLP	LSTM	TCN	TCN-LSTM
1	RMSprop	ReduceLROnPlateau	True	False	10	0.52	0.67	0.30
2	RMSprop	CyclicLR	True	False	54	0.55	0.64	0.25
3	Adam	ReduceLROnPlateau	True	False	17	0.38	0.61	0.13
4	RMSprop	ReduceLROnPlateau	False	False	10	0.52	0.62	0.28
5	RMSprop	CyclicLR	False	False	10	0.53	0.66	0.27
6	Adam	ReduceLROnPlateau	False	False	13	0.61	0.74	0.26
7	RMSprop	ReduceLROnPlateau	False	True	44	0.55	0.48	0.12
8	RMSprop	CyclicLR	False	True	10	0.66	0.73	0.27
9	Adam	ReduceLROnPlateau	False	True	121	0.44	0.49	0.16
10	RMSprop	ReduceLROnPlateau	True	True	158	0.72	0.73	0.17
11	RMSprop	CyclicLR	True	True	11	0.69	0.70	0.18
12	Adam	ReduceLROnPlateau	True	True	19	0.67	0.65	0.14

Figure 8: Order probability Vs. Add to cart order

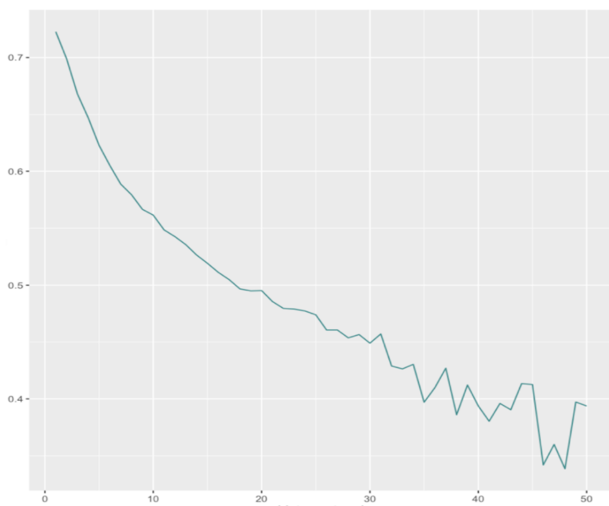


Table 4: BCELoss of Test2 for 6 best Trials of ML Models

Trial	Hyper-Parameter	Xgboost	RandomForest
1	Bayesian-Optimizer	0.18	0.18
2	Bayesian-Optimizer	0.18	0.18
3	Bayesian-Optimizer	0.18	0.18
4	Bayesian-Optimizer	0.18	0.18
5	Bayesian-Optimizer	0.18	0.18
6	Bayesian-Optimizer	0.18	0.18

Table 5: Training Results

Model Type	Train BCELoss	Val BCELoss	Test1 BCELoss	Test2 BCELoss
MLP	0.18	0.18	0.18	0.18
LSTM	0.18	0.17	0.17	0.17
TCN	0.18	0.15	0.15	0.15
TCNLSTM	0.18	0.13	0.13	0.13
Xgboost	0.18	0.21	0.21	0.21
RandomForest	0.18	0.23	0.23	0.23

Table 6: Stacked Generalization Results

Model Type	K Value	Train BCELoss	Val BCELoss	Test1 BCELoss	Test2 BCELoss
Weighted K Best	5	0.18	0.18	0.18	0.18
Weighted K Best	10	0.18	0.18	0.18	0.18
Weighted K Best	25	0.18	0.18	0.18	0.18
K Best	3	0.18	0.18	0.18	0.18
K Best	5	0.18	0.18	0.18	0.18

Table 7: Final Accuracy post F_1-Maximization

Data Split	Precision	Recall	F_1-Score
Train	0.18	0.18	0.18
Validation	0.18	0.18	0.18
Test1	0.18	0.18	0.18
Test2	0.18	0.18	0.18