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

Abstract

Predicting future consumer behaviour is one of the most challenging problems for large scale retail firms. Accurate prediction of consumer purchase pattern enables better inventory planning and efficient personalized marketing strategies. Optimal inventory planning helps minimise instances of Out-of-stock/ Excess Inventory and, smart Personalized marketing strategy ensures smooth and delightful shopping experience. Consumer purchase prediction problem has generally been addressed by ML researchers in conventional manners, either through recommender systems or traditional ML approaches. Such modelling approaches do not generalise well in predicting consumer purchase pattern. In this paper, we present our study of consumer purchase behaviour, wherein, we establish a data-driven framework to predict whether a consumer is going to purchase an item within a certain time frame using e-commerce retail data. To model this relationship, we create a sequential time-series data for all relevant consumer-item combinations. We then build generalized non-linear models by generating features at the intersection of consumer, item, and time. We demonstrate robust performance by experimenting with different neural network architectures, ML models, and their combinations. We present the results of 60 modelling experiments with varying Hyperparameters along with Stacked Generalization ensemble [25] and F_1-Maximization framework. We then present the benefits that neural network architectures like Multi Layer Perceptron, Long Short Term Memory (LSTM), Temporal Convolutional Networks (TCN) [13] and TCN-LSTM [12] bring over ML models like Xgboost [4] and RandomForest.

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 [5] 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 don't 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 [21], and item-affinity remain unaccounted for. Such inefficiencies lead to a gap between consumer needs and item availablity, 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 [25] 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, usefullness 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 [6, 5]. Some previous studies have analyzed the use of random forest and Xgboost

techniques in order to predict consumer retention, where past consumer behavior was used as potential explanatory variable for modelling such patterns. In one such study [15], 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 [9] which outperform the gradient boosting type of models like Xgboost [4].

From Neural Network architectures perspective, close to our work is Deep Neural Network Ensembles for Time Series Classification [8]. 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 [20], Multi-Channels Deep Convolutional Neural Networks [29] and Time Convolutional Neural Network [28]. The first three were originally proposed in [24]. We propose the application of such architectures in the consumer choice world and apply the concept of entity embeddings [9] along with neural network architectures like Multi Layer Perceptron, Long Short Term Memory (LSTM), Temporal Convolutional Networks (TCN) [13] and TCN-LSTM [12].

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. *Postive 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 4 parts - train, validation, test1, and test2 as shown in Table 1. 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.

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
Test1	Stacked Generalization, F_1-Maximization	50,899	2 weeks
Test2	Reporting Accuracy Metrics	50,910	2 weeks

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

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

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 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:

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

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.

3.2 Model Architectures

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 of the features. Hence, we work with tree based models like RandomForest, Xgboost [4] to Deep learning models like Multi Layer Perceptron (MLP), Long Short Term Memory (LSTM) and Temporal Convolutional Networks (TCN). Architectures of MLP, LSTM, TCN [13] and TCN-LSTM [12] models are shown in Figure 2, Figure 3, Figure 4 and Figure 5 respectively. We briefly describe the architectures below.

- **Entity Embeddings + Multi Layer Perceptron:** MLP (Figure 2) is the simplest form of deep neural networks and was originally proposed in [24]. The architecture contains three hidden layers fully connected to the output of its previous layer. The final layer is the sigmoid layer which generates the probability. One disadvantage of MLP is that since the input time series is fully connected to the first hidden layer, the temporal information in a time series is lost [7].
- **Entity Embeddings + Long Short Term Memory:** LSTM (Figure 3) is an architecture comprising of 2 LSTM layers combined with entity embeddings. This combination flows into 3 fully connected ReLU based layers yielding to dense layer which has sigmoid activation.
- **Entity Embeddings + Temporal Convolutional Network:** TCN (Figure 4), originally proposed in [13] , 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, it flows into 3 fully connected ReLU based layers yielding to dense layer which has sigmoid activation.
- **Entity Embeddings + Long Short Term Memory-Temporal Convolutional Network:** TCN-LSTM (Figure 5) inherits the properties of LSTM and TCN in a fully connected network.

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.

As mentioned earlier in this section, in all the above described neural network architectures, we learn the embeddings [9] of the categorical features during training phase. We embed these attributes in order to compress their representations while preserving salient features, and capture mutual similarities and differences.

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 variations, it becomes imperative to cross learn consumer behaviour from

Figure 1: Data classification for DNN Architectures

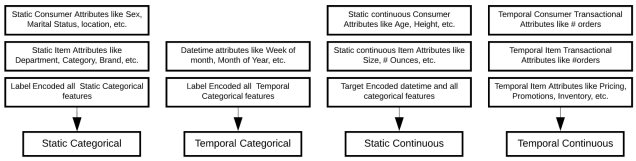
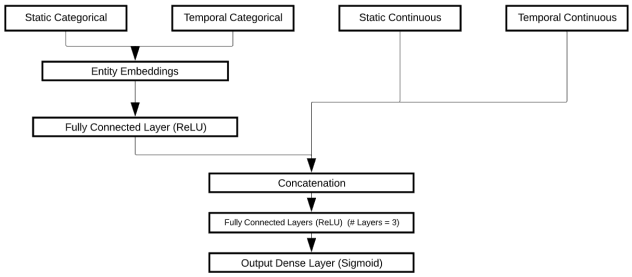


Figure 2: Multi Layer Perceptron (MLP)



similar consumer groups. To learn such relationships its important to capture non-linear relationship between target and regressors at various levels. Tree based and Deep learning models are chosen for their ability to model feature interactions (even if transient in time), and their ability to capture non-linearity well.

Tree based models and MLP training uses lagged values of time varying regressors to capture temporal dependencies. LSTM, TCN and TCN-LSTM models are trained using entire life-cycle data of a time series (Consumer-Item) in sequential manner. Details around dataset and features are explained in section 4.

3.3 Hyperparameter Tuning

Hyper-parameters of tree based models are optimized using Bayesian Hyperparameter Optimization Technique, Hyperopt [2]. 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 [1] 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 [22] and ReduceLROnPlateau [27] 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) [11] is used to improve generalization across Deep Learning models. SWA performs an equal average of the weights traversed

Figure 3: Long Short Term Memory (LSTM)

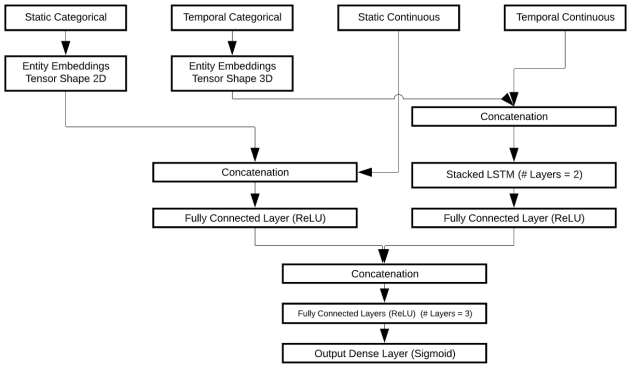
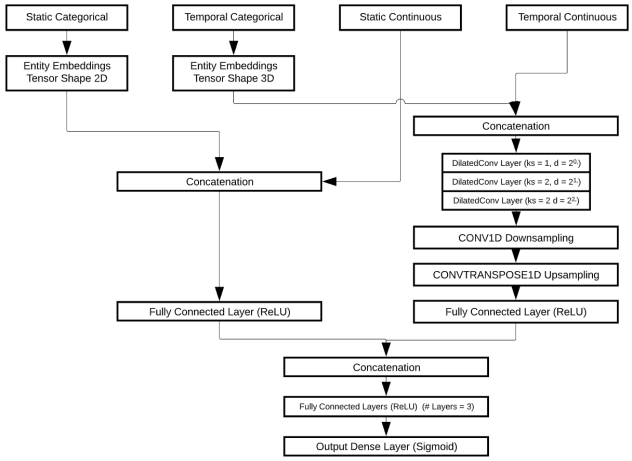


Figure 4: Temporal Convolutional Network (TCN)



by SGD with a modified learning rate schedule. We used 1e-3 as SWA learning rate.

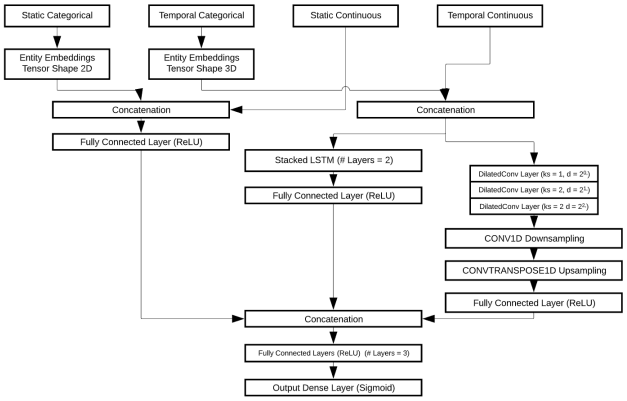
- **Parameter Average:** This is another generalization technique, which is weighted average of parameter weights of n minimas post training completion. Weights are based on total val loss for epoch with n set as 2 for our experiments.

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. For sequence model [23], we also iterate with Dropout [10] and BatchNorm [19] within networks. Hyperparameters used for Machine learning models with Hyperopt [2] are :

- **Learning Rate:** Range set to vary between 1e-2 to 5e-1.
- **Max Depth:** Range set from 2 to 12 at step of 1.

Apart from above hyperparameters, Regularization parameters like Reg Lambda and Min Sample Leaf were also optimized using Hyperopt.

Figure 5: TCN-LSTM



Deep learning models are built using deep learning framework PyTorch [16], and are trained on GCP instance containing 6 CPUs and a single GPU. Scikit-learn [17] is used for Tree based models like RandomForest and Xgboost [4]. For Neural Network Architectures, we save model weights of the best checkpoint, so that we may access the learned entity embeddings and other model weights in case of requirement. As described in Table 2, we build a total of 60 models, 12 different configurations for each of 4 Deep Learning models and 6 best trials from Hyperopt [2] for each of 2 Machine Learning models.

3.4 Stacked Generalization Ensemble

Stacked generalization or Stacking [25] is an ensemble method where a new model learns how to best combine the predictions from multiple existing models. This is acieved by training an entirely new model using contributions from each submodel. We use weighted K-best model as Stacker for combining k submodels (candidates) out of total 60 trained models. We iterate with different values of k ranging from 3 to 25 as presented in Table 6 . Test1 BCELoss of submodels is used for weight initialization for the stacker models. For learning optimal weights of submodels, we minimise Test1 BCELoss of the stacker model using gradient descent [18], stacking function can be described as:

$$y_{\text{cit}} = \sum_{j=1}^{k} w_{\text{j}} \times p_{\text{cit}_{\text{j}}} \quad (3)$$

where y_{cit} is the stacked probability for consumer 'c' for item 'i' at time 't'. k represents the number of candidates shortlisted for stacking, $p_{\text{cit}_{\text{j}}}$ represents the prediction probability for consumer 'c' for item 'i' at time 't' by j^{th} submodel. w_{j} is the weight for j^{th} submodel.

3.5 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 [14]. To illustrate the above, let's say we generated purchase probabilities for 'n' items out of 'b' actually purchased items by

consumer 'c'. Now, lets visualize the actuals (4) and predictions (5) of 'n' predicted items for consumer 'c'.

$$A_c = [a_1, a_2, .., a_n] \; \forall \; a_j \in \{0,1\} \tag{4}$$

$$P_c = [p_1, p_2, .., p_n] \; \forall \; p_j \in \; [0, 1] \tag{5}$$

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' is the total items model generated purchase probabilities for consumer 'c'. Now we apply Decision rule D() which converts probabilities to binary predictions, as described below in Equation 6.

$$D(Pr_c) : {P_c}^{1 \times n} \rightarrow {P'_c}^{1 \times n} \quad : p'_j = \begin{cases} 1 & p_j \geq Pr_c \\ 0 & \text{Otherwise} \end{cases} \tag{6}$$

$$P'_c = [p'_1, p'_2, .., p'_n] \; \forall \; p'_j \in \{0,1\} \tag{7}$$

$$k = \sum_{i=1}^{n} p'_i \tag{8}$$

Pr_c is the probability cut-off to be optimized for maximizing F_1 measure [14] 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.

$$V_{Pr_c} = P'_c \; \times \; {A_c}^T \; \Rightarrow \; \begin{pmatrix} p'_1 & .. & p'_n \end{pmatrix} \times \begin{pmatrix} a_1 \\ .. \\ a_n \end{pmatrix} \tag{9}$$

$$Precision_c = \frac{V_{Pr_c}}{k} \quad and \quad Recall_c = \frac{V_{Pr_c}}{b} \tag{10}$$

$$F_{1_c} = \frac{2 \times Precision_c \times Recall_c}{Precision_c + Recall_c} \quad \Rightarrow \quad 2 * \frac{V_{Pr_c}}{k+b} \tag{11}$$

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 10 and 11 are used to calculate Precision, Recall and F_1-score for consumer 'c'.

$$\max_{V_{Pr_c}} \quad 2 * \frac{V_{Pr_c}}{k+b} \quad , \quad \text{subject to:} \quad Pr_c \in \; [0, 1] \tag{12}$$

Equation 12 represents the optimization function we solve to generate purchase predictions (1/0) for each consumer.

4 Experiments and Results

We use transactional data from instacart kaggle challenge to train all our models (sample data 6). From sample data we can see that data contains transactional details including order id, add to cart order, date of transaction, aisle id and department id for each consumer-item transaction. As described in Table 1, we utilize 1 year data for each consumer-item combination, which then gets split into train, validation, test1 and test2 as per our validation strategy. We generate consumer-item-week level data with purchase/ non purchase being the target, and use this data to train all our models and generate predictions.

4.1 Experiment Setups

We started with exploratary data analysis, looking at the data from various cuts and trying to study the variations of the features with target. Some of them include looking at the density of consumers Vs. basket size Figure 8, reorder visualization of items across departments Figure 7, variation of reorder probability with add to cart order Figure 9, order probability variations at different temporal cuts like week, month and quarter, transactional metrics like total orders, total reorders, recency, gap between orders, at both consumer and item levels. We then performed multiple experiments with the above mentioned features and different hyperparameter configurations to land at reasonable hyperparameters to perform final experiments and present our results.

4.2 Results and Observations

Tables 3 and 4 show the experimental results obtained across models with different hyperparameter configurations. Table 3 contains the Deep Learning Experiment setup results and Table 4 has Machine Learning model results. From model performance perspective, it is observed that TCN showed the best Accuracy score at test2 (out of time), with most of the other Deep Learning models having comparable scores. Also, from Table 5 we see that all the Deep Learning models out perform Machine Learning models including Xgboost and RandomForest both in terms of Accuracy and Generalization. This table has the average of BCELoss across 3 best trials, and it can be observed that TCN has the lowest BCELoss, which translates into best Accuracy. From hyperparameter configuration perspective we observe that RMSprop and CyclicLR emerged as the clear winners as Optimizer and Scheduler respectively (from Table 3). 7 out of 12 times, we found this combination (out of 3 possible combinations) generating the best result.

We also present the effectiveness of combining predictions in the form of stacking. From Table 6, we can see the results of stacking at different values of K, for Weighted K-Best model setups. We realised the best outcome at K = 3, which led to a better score compared to any individual model. In Figure 10, we can see the probability distributions post stacking for both labels of the target. Finally we apply F_1-Maximization over stacked probability values so as to generate binary predictions(1/0 referring to Purchase/Non Purchase). F_1-Score Optimizer helps strike a balance between Precision and Recall [3]. Post F_1-Maximization we observe that Precision, Recall and F_1-Score are close enough for all data splits, as can be seen in Table 7. We scored **0.4109** over unseen data (Test2) from Table 7 as F_1-Score with the described framework.

4.3 Industrial Applications

The Next Logical Purchase framework has multiple applications in retail/e-retail industry. Some of them include:

- **Personalized Marketing:** With prior knowledge of next logical purchase, accurate item recommendations and optimal offer rollouts can be made at consumer level. This will enable a seamless and delightful consumer shopping experience.

Table 2: Model Specifications

Model Type	Trials	Model HyperParameters	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	BCELoss
RandomForest	6	Tree Depth, Evaluation Metrics, Regularization parameters	BCELoss

Figure 6: 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

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	**0.0276**	0.0306	**0.0249**	0.0307
2	RMSprop	CyclicLR	True	False	0.0708	**0.0269**	**0.0269**	0.0348
3	Adam	ReduceLROnPlateau	True	False	0.0295	0.0303	0.0667	0.0337
4	RMSprop	ReduceLROnPlateau	False	False	0.0297	**0.0275**	0.0364	0.0759
5	RMSprop	CyclicLR	False	False	**0.0250**	0.0306	0.0600	**0.0286**
6	Adam	ReduceLROnPlateau	False	False	0.0360	**0.0302**	0.0590	0.0309
7	RMSprop	ReduceLROnPlateau	False	True	0.0293	0.0432	0.0453	0.0381
8	RMSprop	CyclicLR	False	True	**0.0245**	0.0378	0.0569	**0.0262**
9	Adam	ReduceLROnPlateau	False	True	0.0700	0.0491	0.0610	0.0382
10	RMSprop	ReduceLROnPlateau	True	True	0.0356	0.0364	**0.0238**	0.0309
11	RMSprop	CyclicLR	True	True	0.0420	0.0377	0.0284	**0.0269**
12	Adam	ReduceLROnPlateau	True	True	0.0321	0.0306	0.0547	0.0305

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

Trial	HyperParameter	Xgboost	RandomForest
1	HyperOpt	**0.0332**	0.0526
2	HyperOpt	0.0364	0.0479
3	HyperOpt	0.0347	**0.0416**
4	HyperOpt	0.0364	**0.0449**
5	HyperOpt	**0.0335**	**0.0459**
6	HyperOpt	**0.0339**	0.0578

- **Inventory Planning:** Short Term Inventory Planning (2-4 weeks) is largely dependant over consumer preference in near future. Consumer preference model can be used in better short term inventory planning.
- **Assortment Planning:** In retail stores, consumer choice study can be used to optimize the store layout with right product placement over right shelf.

Table 5: BCELoss mean of top 3 trials across data splits

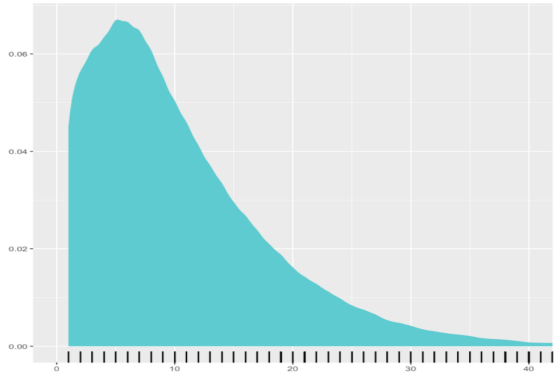
Model Type	Val BCELoss	Test1 BCELoss	Test2 BCELoss
MLP	0.0405	0.0289	0.0256
LSTM	0.0373	0.0293	0.0282
TCN	**0.0368**	**0.0292**	**0.0251**
TCNLSTM	0.0368	0.0304	0.0273
Xgboost	0.0352	0.0318	0.0335
RandomForest	0.0437	0.0389	0.0441

5 Conclusion

We have presented our study of the consumer purchase behaviour in the context of large scale e-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

Figure 7: Most reordered Items across Departments

Figure 8: Density of consumers Vs. Basket Size



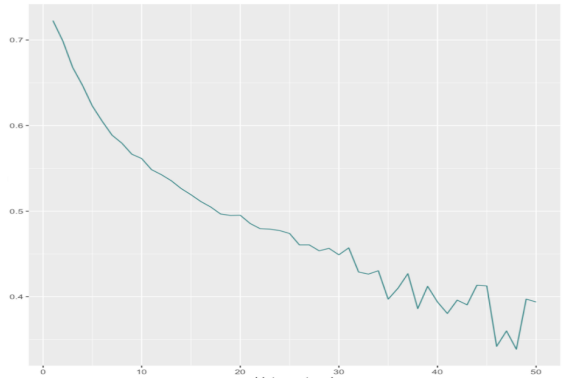
Basket size between 5-8 has maximum consumer density

might not be available as is when predictions are being generated for future period. Hence we have saved model weights and used innovative transformations so that we don't necessarily have to remember complete 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, and our results and observations were inline with this thought process. Model generalization and robustness was attained using stacked generalization. As per our expectation we realized gain in accuracy post both stacking and F_1 - Maximization.

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 [26]. Also, we are working, to further improve our Sequence to Sequence neural network architectures to improve accuracy and decrease computation time.

Figure 9: Reorder probability Vs. Add to cart order



Probability of reordering decreases as the order of add to cart increases

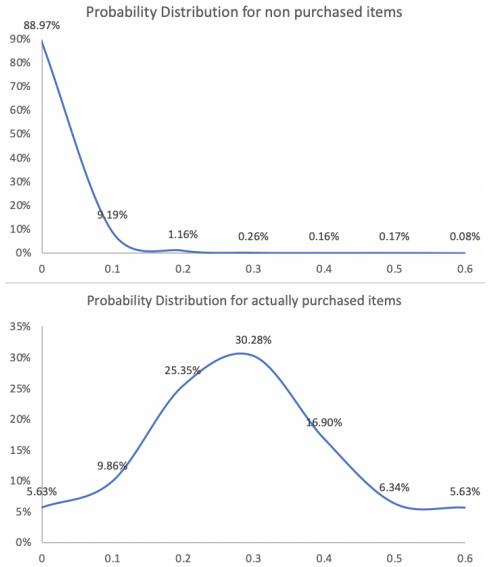
Table 6: Stacked Generalization Results

Model Type	K Value	Val BCELoss	Test1 BCELoss	Test2 BCELoss
Weighted K Best	**3**	**0.0386**	**0.0278**	**0.0242**
Weighted K Best	5	0.0373	0.0282	0.0245
Weighted K Best	10	0.0397	0.0290	0.0258
Weighted K Best	15	0.0389	0.0296	0.0272
Weighted K Best	25	0.0394	0.0316	0.0287

Table 7: Final Accuracy post F_1-Maximization

Data Split	Precision	Recall	F_1-Score
Validation	0.3401	0.4981	0.4042
Test1	0.3323	0.5103	0.4024
Test2	0.3506	0.4964	0.4109

Figure 10: Probability Distributions



High density probability zone for the non purchased cases lies between 0 and 0.1, whereas for the purchased cases it lies between 0.25 and 0.35

References

[1] Bengio, Y.; and CA, M. 2015. Rmsprop and equilibrated adaptive learning rates for nonconvex optimization. *corr abs/1502.04390* .

[2] Bergstra, J.; Yamins, D.; and Cox, D. D. 2013. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in science conference*, volume 13, 20. Citeseer.

[3] Buckland, M.; and Gey, F. 1994. The relationship between recall and precision. *Journal of the American society for information science* 45(1): 12–19.

[4] Chen, T.; and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

[5] Choudhury, A. M.; and Nur, K. 2019. A Machine Learning Approach to Identify Potential Customer Based on Purchase Behavior. In *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, 242–247. IEEE.

[6] Fader, P. S.; and Hardie, B. G. 2009. Probability models for customer-base analysis. *Journal of interactive marketing* 23(1): 61–69.

[7] Fawaz, H. I.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2019. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery* 33(4): 917–963.

[8] Fawaz, H. I.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2019. Deep neural network ensembles for time series classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–6. IEEE.

[9] Guo, C.; and Berkhahn, F. 2016. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737* .

[10] Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* .

[11] Izmailov, P.; Podoprikhin, D.; Garipov, T.; Vetrov, D.; and Wilson, A. G. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407* .

[12] Karim, F.; Majumdar, S.; Darabi, H.; and Chen, S. 2017. LSTM fully convolutional networks for time series classification. *IEEE access* 6: 1662–1669.

[13] Lea, C.; Vidal, R.; Reiter, A.; and Hager, G. D. 2016. Temporal convolutional networks: A unified approach to action segmentation. In *European Conference on Computer Vision*, 47–54. Springer.

[14] Lipton, Z. C.; Elkan, C.; and Naryanaswamy, B. 2014. Optimal thresholding of classifiers to maximize F1 measure. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 225–239. Springer.

[15] Martínez, A.; Schmuck, C.; Pereverzyev Jr, S.; Pirker, C.; and Haltmeier, M. 2020. A machine learning framework for customer purchase prediction in the non-contractual setting. *European Journal of Operational Research* 281(3): 588–596.

[16] Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch .

[17] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12: 2825–2830.

[18] Ruder, S. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* .

[19] Santurkar, S.; Tsipras, D.; Ilyas, A.; and Madry, A. 2018. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, 2483–2493.

[20] Serrà, J.; Pascual, S.; and Karatzoglou, A. 2018. Towards a Universal Neural Network Encoder for Time Series. In *CCIA*, 120–129.

[21] Shah, J.; and Avittathur, B. 2007. The retailer multi-item inventory problem with demand cannibalization and substitution. *International journal of production economics* 106(1): 104–114.

[22] Smith, L. N. 2017. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 464–472. IEEE.

[23] Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

[24] Wang, Z.; Yan, W.; and Oates, T. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, 1578–1585. IEEE.

[25] Wolpert, D. H. 1992. Stacked generalization. *Neural networks* 5(2): 241–259.

[26] Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, 3320–3328.

[27] Zaheer, M.; Reddi, S.; Sachan, D.; Kale, S.; and Kumar, S. 2018. Adaptive methods for nonconvex optimization. In *Advances in neural information processing systems*, 9793–9803.

[28] Zhao, B.; Lu, H.; Chen, S.; Liu, J.; and Wu, D. 2017. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics* 28(1): 162–169.

[29] Zheng, Y.; Liu, Q.; Chen, E.; Ge, Y.; and Zhao, J. L. 2014. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, 298–310. Springer.