

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

Ankur Verma

ankur.verma.phe09@iitbhu.ac.in

## 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 Hyper-parameters along with Stacked Generalization ensemble [22] 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) [11] and TCN-LSTM [10] bring over ML models like Xgboost [3] and RandomForest.

## 1 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, sales and marketing are some of the key factors driving today's consumer world in retail. While very little information is available on future variabilities of the above factors, what retailers have is large volumes of transactional data. Retailers use conventional techniques with the available data to model consumer purchase [4]. 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.

In many of the retail/e-retail brands, short term (2-4 weeks ahead) inventory planning is done on the basis of consumer purchase pattern. Also, certain sales and marketing strategies like Offer Personalization, 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 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 cannibalization [18], and item-affinity remains unaccounted. Such inefficiencies lead to gap between consumer needs and item availability, resulting in the 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 item purchase at consumer level. We showcase the performance of individual models with varying hyper-parameter configurations along with the results of stacked generalization ensemble [22] (algorithmic combination of predictions from different models) and  $F_1$ -maximization (optimal purchase probability cut-off at consumer level). In the next section 2, we briefly discuss research work related to the problem in hand. Section 3 explains the overall methodology adopted to solve the problem. It lays out and neural network architectures and various algorithmic variants applied to the problem. Section 4 describes the experiments performed, and results obtained in different modelling setups.

## 2 Related Work

In the past few years, 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 those approaches are based on extracting consumer's latent characteristics from its past purchase behavior and applying statistical and ML based formulations [5, 4]. Some of 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 [13], 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 [7] which outperforms the gradient boosting type of models like Xgboost [3].

Over Time Series Classification part of our problem, Close to our work is Deep Neural Network Ensembles for Time Series Classification [6]. 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 [17], Multi-Channels Deep Convolutional Neural Networks [25] and Time Convolutional Neural Network [24]. The first three were originally proposed in [21]. We apply the concept of entity embeddings [7] along with neural network architectures like Multi Layer Perceptron, Long Short Term Memory (LSTM), Temporal Convolutional Networks (TCN) [11] and TCN-LSTM [10], and prove their application in modelling consumer-item time series data.

### 3 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.
- **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

Table 1: Modelling data splits

Data Split	Specifications	Consumer-Item combinations	Max Time-Series length
Train	Model training	50,872	46 weeks
Validation	Hyper-Parameter Optimization	50,888	2 weeks
Test1	Stacked Generalization, $F_1$ -Maximization	50,899	2 weeks
Test2	Reporting Accuracy Metrics	50,910	2 weeks

take into account consumer and item variations in comparing the time series. A row in time series is represented by

$$y_{cit} = f(i_t, c_t, \dots, c_{t-n}, i_{c_t}, \dots, i_{c_{t-n}}, d_t, \dots, d_{t-n}) \quad (1)$$

where  $y_{cit}$  is purchase prediction 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.  $i_{c_t}$  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.

#### 3.1 Loss Function

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

$$H_p = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)) \quad (2)$$

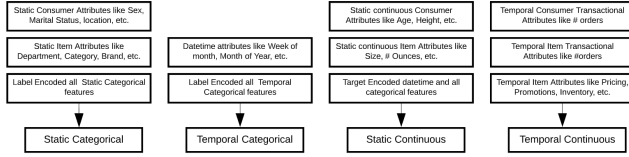
where  $H_p$  is total loss and  $y_i$  is the label and  $p(y_i)$  is the predicted probability.

#### 3.2 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 RandomForest, Xgboost [3] 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 [11] and TCN-LSTM [10] models are shown in Figure 2, Figure 3, Figure 4 and Figure 5.

- **Entity Embeddings + Multi Layer Perceptron:** (MLP) Figure 2 is the simplest form of deep neural networks and was proposed in [21] as a baseline architecture for Time Series classification. The architecture contains three hidden layers fully connected to the output of its previous layer. The final layer is the sigmoid layer to generate probabilities. One disadvantage is that since the input time series is fully connected to the first hidden layer, the temporal information in a time series is lost [6].
- **Entity Embeddings + Long Short Term Memory:** (LSTM) Figure 3 is a sequence to sequence [20] 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 [11], is considered a competitive architecture yielding the best results when evaluated over our experimental dataset. This

Figure 1: Data classification for DNN Architectures



network is comprised of 3 dilated Convolutional network combined with entity embeddings. Similar to LSTM, it is also a sequence to sequence architecture which, after convolving and concatenation 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 is again sequence to sequence architecture which inherits the properties of LSTM and TCN in a fully connected network.

As described in architecture diagrams Figure 1, data was classified into following groups:

- **Static Categorical:** These are categorical features which donot vary with time. This includes the consumer attributes like sex, marital status and location, different item attributes like category, department and brand.
- **Temporal Categorical:** These are categorical features which 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 class.

In all the above neural network architectures, we learn the embeddings [7] of the categorical features during training phase. We embed these attributes in order to both compress their representations while preserving salient features, as well as 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 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

Figure 2: Multi Layer Perceptron (MLP)

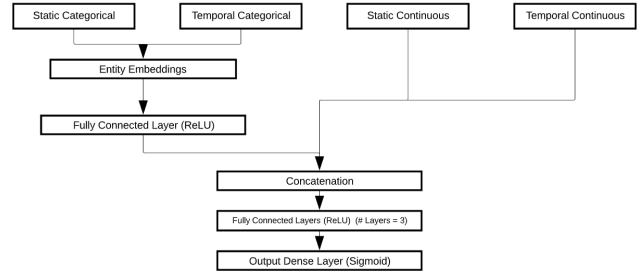
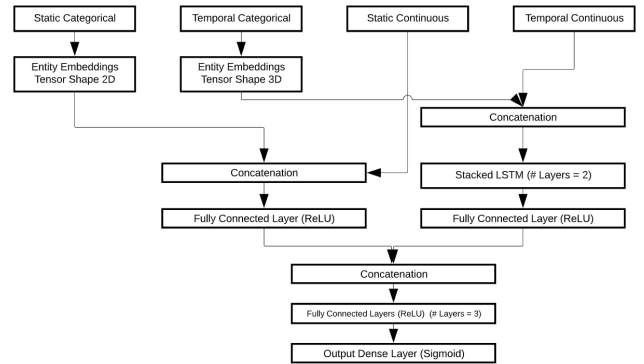


Figure 3: Long Short Term Memory (LSTM)



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 Hyperparameter Optimization Technique [2]. LSTM, TCN and TCN-LSTM models were trained in sequence to sequence fashion [20] 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 [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.

Figure 4: Temporal Convolutional Network (TCN)

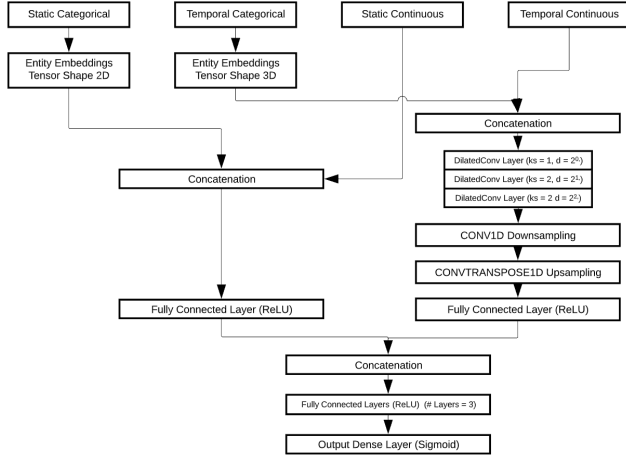
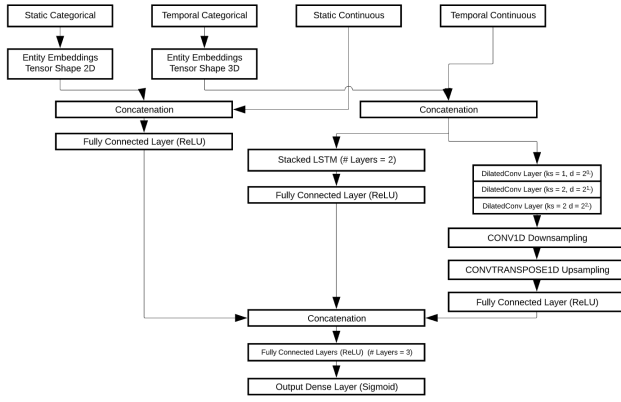


Figure 5: TCN-LSTM



- **Scheduler Parameters:** Cyclic [19] and ReduceLROnPlateau [23] 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) [9] 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.
- **Parameter Average:** This applies weighted average over the parameter weights of n best performing epochs (local minimas) from state dictionary post completion of training.

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. We have observed an improved performance of the sequence model [20] when Dropout [8] and BatchNorm [16] are used

We used Bayesian Hyper-parameter Optimization [2] for hyper configuration tuning with 100 trials. Some of the hyperparameters we used for Machine learning models are :

- **Learning Rate:** LR range was set to vary between 1e-3 to 0.5.
- **Max Depth:** Depth range was set between 2 to 12 at the step of 1.

Apart from these, Regularization parameters like Reg Lambda, Min Sample Leaf was also tuned from Bayesian framework.

Deep learning models are built using deep learning framework PyTorch [14], and are trained on GCP instance containing 6 CPUs and a single GPU. scikit-learn [15] is used for Tree based models like RandomForest and Xgboost [3]. 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 2.

### 3.3 Stacked Generalization

For simplicity and better generalization we adopted non-parametric approach for Stacked Generalization [22] . We used Weighted K-Best and K-Best as Stacked Generalization Ensemble model for combining the 60 models (candidates) from Table 2. 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. A row post stacking is represented as

$$y_{cit} = \sum_{j=1}^k w_j \times p_{cit_j} \quad (3)$$

where  $y_{cit}$  is the stacked probability for consumer 'c' item 'i' at time 't'.  $w_k$  is the candidate weight of  $k^{th}$  model. k represents the number of candidates shortlisted for stacking and  $p_{cit_k}$  represents the prediction probability for consumer 'c' item 'i' at time 't' by  $k^{th}$  model.

### 3.4 $F_1$ -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 using  $F_1$ -Measure [12]. 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  $[Ia_1, Ia_2, \dots, Ia_{n_i}]$  whereas items for which the model generated probabilities for the consumer  $c_i$  at test1 time step be  $[Ip_1, Ip_2, \dots, Ip_{n_i}]$ .

$$A_{c_i} = [a_1, a_2, \dots, a_{n_i}] \forall a_j \in \{0,1\} \quad (4)$$

$$P_{c_i} = [p_1, p_2, \dots, p_{n_i}] \forall p_j \in [0,1] \quad (5)$$

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	BCELoss
RandomForest	6	Tree Depth, Evaluation Metrics, Regularization parameters	BCELoss

$A_{c_i}$  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.

$$D(P_{c_i}) : P_{c_i}^{1 \times n_i} \rightarrow P'_{c_i}^{1 \times n_i} : p'_j = \begin{cases} 1 & p_j \geq Pr_{c_i} \\ 0 & p_j < Pr_{c_i} \end{cases} \quad (6)$$

$$P'_{c_i} = [p'_1, p'_2, \dots, p'_{n_i}] \forall p'_j \in \{0,1\} \quad (7)$$

$$k_i = \sum_{j=1}^{n_i} p'_j \quad (8)$$

$Pr_{c_i}$  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.  $k_i$  is the total number of predictions which Decision rule  $D$  converted to 1 or in other words number of predictions which model predicted as purchase.

$$V_{Pr_{c_i}} = P'_{c_i} \times A_{c_i}^T \Rightarrow (p'_1 \dots p'_{n_i}) \times \begin{pmatrix} a_1 \\ \vdots \\ a_{n_i} \end{pmatrix} \quad (9)$$

$V_{Pr_{c_i}}$  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  $c_i$ .

$$Precision_{c_i} = \frac{V_{Pr_{c_i}}}{k_i} \quad \text{and} \quad Recall_{c_i} = \frac{V_{Pr_{c_i}}}{b_i} \quad (10)$$

$$F_{1_{c_i}} = \frac{2 \times Precision_{c_i} \times Recall_{c_i}}{Precision_{c_i} + Recall_{c_i}} \Rightarrow 2 * \frac{V_{Pr_{c_i}}}{k_i + b_i} \quad (11)$$

$F_{1_{c_i}}$  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.

## 4 Experiments and Results

We use transactional data from instacart kaggle challenge to train all our models. As can be seen in Figure 6 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 forwards 2 time steps (2 weeks for our case).

### 4.1 Experiment Setups

We started with exploratory 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 with different basket sizes Figure 8, orders placed for items across departments Figure 7, variation of order probability with add to cart order Figure 9, order probability variations at different temporal cuts like week, month and quarter, transactional metrics at both consumer and item levels like total orders, total reorders, recency, gap between orders, etc. We then performed multiple experiments with above features and different hyper-configurations.

### 4.2 Results and Observations

Tables 3 and 4 shows the experimental results obtained with different hyper-configurations over multiple models. From Table 3, we can infer that RMSprop and CyclicLR emerged as the clear winners as Optimizer and Scheduler respectively. TCN showed the best Accuracy score at test2, with most of the other Deep Learning models have comparable results. Also, we observe that all the Deep Learning models out perform Machine Learning models including Xgboost and RandomForest both in terms of Accuracy and Generalization. This insight can be drawn from Table 5 which has the average of BCELoss across 3 best trials. Range of scores across val, test1 and test2 states that Deep Learning models are showing more stable results in terms of model fit. We 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  $K$ , both for Weighted K-Best as well as K-Best model setups. Finally we apply  $F_1$ -Maximization so as to be able to strike a balance between Precision and Recall. After  $F_1$ -Maximization we see that all Precision, Recall and  $F_1$ -Score are close enough for all data splits as can be seen from Table 7.

### 4.3 Industrial Applications

The Next Logical Purchase framework has multiple applications in retail industry. Some of its applications include

- **Personalized Marketing:** Item Recommendations at consumer level can be derived from the solution of this problem, which will enhance consumer shopping experience.
- **Inventory Planning:** This can help in Planning inventory well, as it provides the consumer choice at a point time.

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

Trial	Optimizer	Scheduler	SWA	Parameter Avg	MLP	LSTM	TCN	TCN-LSTM
1	RMSprop	ReduceLROnPlateau	True	False	<b>0.0276</b>	0.0306	<b>0.0249</b>	0.0307
2	RMSprop	CyclicLR	True	False	0.0708	<b>0.0269</b>	<b>0.0269</b>	0.0348
3	Adam	ReduceLROnPlateau	True	False	0.0295	0.0303	0.0667	0.0337
4	RMSprop	ReduceLROnPlateau	False	False	0.2427	<b>0.0275</b>	0.0364	0.0759
5	RMSprop	CyclicLR	False	False	<b>0.0250</b>	0.0306	0.0600	<b>0.0286</b>
6	Adam	ReduceLROnPlateau	False	False	0.0360	<b>0.0302</b>	0.0590	0.0309
7	RMSprop	ReduceLROnPlateau	False	True	0.2903	0.0432	0.0453	0.0381
8	RMSprop	CyclicLR	False	True	<b>0.0245</b>	0.0378	0.0569	<b>0.0262</b>
9	Adam	ReduceLROnPlateau	False	True	0.0700	0.0491	0.0610	0.0382
10	RMSprop	ReduceLROnPlateau	True	True	0.0356	0.0364	<b>0.0238</b>	0.0309
11	RMSprop	CyclicLR	True	True	0.0420	0.0377	0.0284	<b>0.0269</b>
12	Adam	ReduceLROnPlateau	True	True	0.0321	0.0306	0.0547	0.0305

milk	soy lactosefree	yogurt	trail mix snack mix	energy granola bars	chips	frozen breakfast	frozen vegetarian	frozen produce
	other creams		popcorn	sandy candy corns	cookies cakes	frozen meals	frozen reads ice cream	frozen ice cream
eggs	dairy eggs	cheeses	mint gum	pancakes		crackers	nuts seeds dried fruit	ice cream topping
cream	packaged cheese	refrigerated pasteurizing specialty cheeses	fruit vegetable snacks			frozen pizza	appetizers sides	frozen juice desserts
		facial care	poultry count	seafood meat counter	paper goods	laundry	fresh fruit	packaged produce
digestion	female menstrual products	eye care skin care	skin care	poultry meat	seafood	household storage	produce vegetables	produce vegetables
oral hygiene	shave needs	first aid	hats dog sauce	packaged meat	packaged meat	trash bags cleaning products	household cleaning products	fresh herbs
water sealering sparkling water	energy sports drinks	coffee	tofu meat alternatives	prepared meals	bread bakery	corn buns rolls	cereal pastries	baby accessories
refrigerator	beverages juice nectars	lunch meats	fresh deli tenderloins	prepared meats	breakfast bakery	bakery desserts	breakfast cereal	baby clothes
soft drinks	tea				fruit canned goods	fruit canned goods	fruit canned goods	fruit canned goods
spreads	gelatin batter pancakes	honey syrup nectars	white wines	beers coolers	meals beans	soup broth bouillon	indian latin foods	cat food care
preserved pickles pickled goods	pantry condiments	beeing supplies	alcohol spirits	red wines	fresh pasta sauces	instant dry pasta	bulk grains rice	other

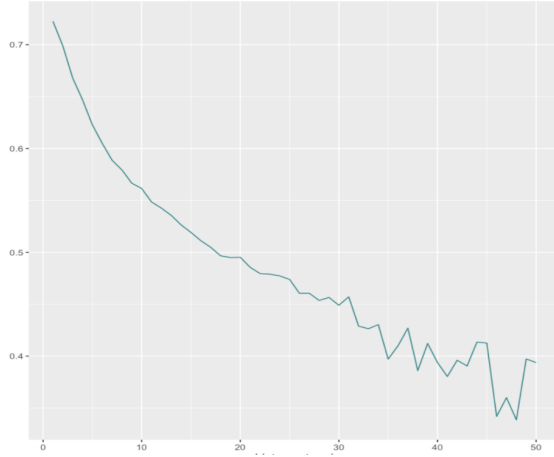
A density plot showing the distribution of the number of children per woman. The x-axis is labeled 'Number of children' and ranges from 0 to 40 with major ticks every 10 units and minor ticks every 2 units. The y-axis is labeled 'Density' and ranges from 0.00 to 0.06 with major ticks every 0.02 units. The plot shows a single, right-skewed distribution peaking at approximately 5 children with a density of about 0.065. The area under the curve is shaded in a light blue color.

- **Assortment Planning:** In retail stores, this can be used for placing items in MOD based on the consumer preference model at a given time frame.

We have presented the study of the Consumer behaviour with Deep Neural Networks and shown how DNNs outper-

form the Machine Learning models like Xgboost and RandomForest. We also showcase the potential of stacked generalisation as well as  $F_1$ -Maximization.

Figure 9: Reorder probability Vs. Add to cart order



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

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

Trial	Hyper-Parameter	Xgboost	RandomForest
1	HyperOpt	<b>0.0332</b>	0.0526
2	HyperOpt	0.0364	0.0479
3	HyperOpt	0.0347	<b>0.0416</b>
4	HyperOpt	0.0364	<b>0.0449</b>
5	HyperOpt	<b>0.0335</b>	<b>0.0459</b>
6	HyperOpt	<b>0.0339</b>	0.0578

## 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] 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.
- [4] 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.
- [5] Fader, P. S.; and Hardie, B. G. 2009. Probability models for customer-base analysis. *Journal of interactive marketing* 23(1): 61–69.
- [6] 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.
- [7] Guo, C.; and Berkhahn, F. 2016. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*.
- [8] 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*.
- [9] Izmailov, P.; Podoprikin, D.; Garipov, T.; Vetrov, D.; and Wilson, A. G. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.
- [10] Karim, F.; Majumdar, S.; Darabi, H.; and Chen, S. 2017. LSTM fully convolutional networks for time series classification. *IEEE access* 6: 1662–1669.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] 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.

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

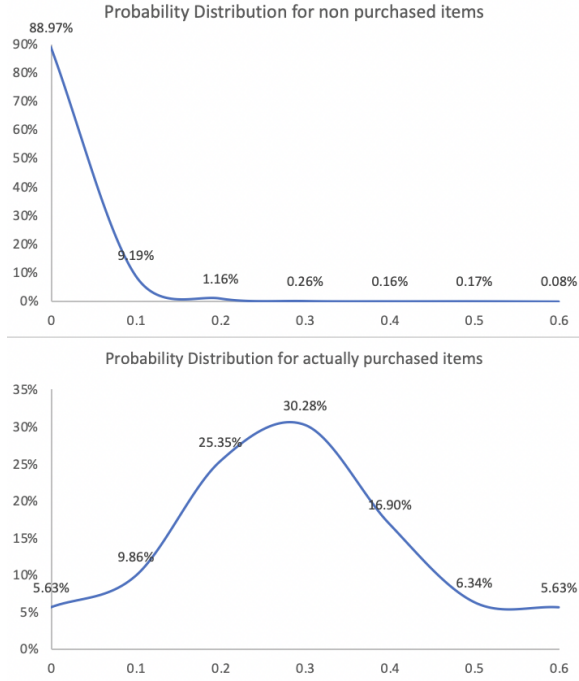
Model Type	Val BCELoss	Test1 BCELoss	Test2 BCELoss
MLP	0.0405	0.0289	0.0256
LSTM	0.0373	0.0293	0.0282
<b>TCN</b>	<b>0.0368</b>	<b>0.0292</b>	<b>0.0251</b>
TCNLSTM	0.0368	0.0304	0.0273
Xgboost	0.0352	0.0318	0.0335
RandomForest	0.0437	0.0389	0.0441

Table 6: Stacked Generalization Results

Model Type	K Value	Val BCELoss	Test1 BCELoss	Test2 BCELoss
<b>Weighted K Best</b>	<b>3</b>	<b>0.0386</b>	<b>0.0278</b>	<b>0.0242</b>
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



Figure 10: Probability Distributions



**Probability Distributions for the non purchased cases tends towards 0 and tapers as the probability moves towards 0.5, whereas for purchased cases the distribution witnesses a peak between 0.2 and 0.3 purchased cases 0.5**

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

quence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

- [21] 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.
- [22] Wolpert, D. H. 1992. Stacked generalization. *Neural networks* 5(2): 241–259.
- [23] 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.
- [24] 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.
- [25] 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.

- [16] 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.
- [17] Serrà, J.; Pascual, S.; and Karatzoglou, A. 2018. Towards a Universal Neural Network Encoder for Time Series. In *CCIA*, 120–129.
- [18] 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.
- [19] 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.
- [20] Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Se-