Fashion Retail: Forecasting Demand for New Items

Pawan Kumar Singh
pawan.ks@myntra.com
Myntra Designs Pvt. Ltd.
Bangalore, India

Yadunath Gupta
yadunath.gupta@myntra.com
Myntra Designs Pvt. Ltd.
Bangalore, India

Nilpa Jha
nilpa.jha@myntra.com
Myntra Designs Pvt. Ltd.
Bangalore, India

Aruna Rajan
aruna.rajan@myntra.com
Myntra Designs Pvt. Ltd.
Bangalore, India

ABSTRACT

Fashion merchandising is one of the most complicated problems in forecasting, given the transient nature of trends in colours, prints, cuts, patterns, and materials in fashion, the economies of scale achievable only in bulk production, as well as geographical variations in consumption. Retailers that serve a large customer base spend a lot of money and resources to stay prepared for meeting changing fashion demands, and incur huge losses in unsold inventory and liquidation costs [2]. This problem has been addressed by analysts and statisticians as well as ML researchers in a conventional fashion - of building models that forecast for future demand given a particular item of fashion with historical data on its sales. To our knowledge, none of these models have generalized well to predict future demand at an abstracted level for a new design/style of fashion article. To address this problem, we present a study of large scale fashion sales data and directly infer which clothing/footwear attributes and merchandising factors drove demand for those items. We then build generalised models to forecast demand given new item attributes, and demonstrate robust performance by experimenting with different neural architectures, ML methods, and loss functions.

KEYWORDS

time series, machine learning, tree based models, neural networks, LSTM, loss function, demand forecasting, attribute embedding

ACM Reference Format:
Pawan Kumar Singh, Yadunath Gupta, Nilpa Jha, and Aruna Rajan. 2019. Fashion Retail: Forecasting Demand for New Items. In *KDD 2019 Workshop: AI for fashion The fourth international workshop on fashion and KDD, August 2019, Anchorage, Alaska - USA.* , 10 pages. https://doi.org/10.1145/1122445.1122456

KDD 2019 Workshop, August 2019, Anchorage, Alaska - USA

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Forecasting demand for fashion retail is one of the most difficult forecasting problems in the industry, given fast changing consumer tastes, long (> 8 months) design and production cycles, bulk manufacturing for cost efficiency, heavy competition on pricing, and increasing marketing costs. When planning for fashion merchandise, there is very little information available on what will be prevailing fashion in the future, what the competitor's mix will be, and how particular pricing and marketing interventions may need to be applied to promote merchandise. What retailers have is large volumes of previous years' sales data and they use it to forecast future purchases using conventional techniques [11]. While these help in estimating demand at reasonable levels of confidence for existing/previously sold merchandise, they cannot be used for predicting demand for new merchandise. Since multiple parameters in design interact non-linearly to define the look or appeal of an item in fashion, past sales data in itself is not instructive in predicting demand for future designs.

In many fashion houses or retail brands, demand planning for the next season (6 months ahead) is done by merchandisers based on their reading of the market, several visits of production and design houses, and their personal observations of what people buy. There is high variability in choices that different buyers recommend, and being limited by intuition, buyers cannot make futuristic calls on price movements and competition pressure. Besides this, every buyer works on a narrow segment of the overall fashion merchandise (such as women's cotton kurtas), and two buyers do not interact or compare merchandise forecasts to adjust their overall forecasts. Hence, effects like product substitution, cannibalization, price-wars between different articles fulfilling the same consumer need, etc cannot be foreseen or accounted correctly. Such inefficiencies lead to significant mismatch in the supply and demand, thus resulting in loss of business opportunity for some items, and piles of unsold inventory (working capital loss). Other than business losses, unsold inventory also leads to considerable environmental damage due to overproduction as well as disposal of unsold inventory. Hence, accurate demand forecasting well into the future of 6-8 months is crucial for better environmental health and business health.

In this paper, we apply deep learning and tree based machine learning algorithms to get point estimates in forecasting demand for items which were not present in the catalog earlier (new or unseen items). In the next section, we briefly discuss research work

related to the current problem. Section 3 explains various algorithmic variants and neural network architectures applied to the problem. Section 4 describes the data used for experiments, and the results obtained in various scenarios of modeling as well as real world deployments.

2 RELATED WORK

Traditionally, time series forecasting has been the tool-set of choice for forecasters and statisticians in retail. These models assume a continuous scenario, where historic patterns are projected into the future. For articles yet to be introduced in fashion, these methods do not hold water. Simpler methods of projecting new and unseen articles are discussed in [11] such as average forecast, seasonal forecast, bass model, life cycle approach, etc. The bass model is an interesting diffusion based model that relies on all products having early adopters (innovators) and late ones (imitators), while the product persists for a longer duration. Products in fashion retails are neither durable nor do they have enough life to have innovators and imitators, thus, making this model inapplicable in our scenario. We use the average forecast model as a baseline to calculate and contrast our model's performance. A comprehensive survey of demand forecast in fashion is reviewed in [12], however, this does not talk about forecasting demand for new items.

New item forecast was first proposed in [18], which uses clustering of past items' sales curve followed by assigning existing items according to a tree based model to one of the clusters. The average sales curve of the cluster is assumed to be the sales for the new items. In our efforts to reuse this method on our fashion sales data, we found that all the items which went live on our platform at the same time grouped together, irrespective of their attributes, price and discounting. We also noticed a lack of similarity in sales behaviour of similarly clustered items even by design attributes, and visual similarity. This is intuitively justifiable, because its the combination of pricing, brand, and relative placement of a certain design which plays on a customer's mind much more heavily than either of them alone.

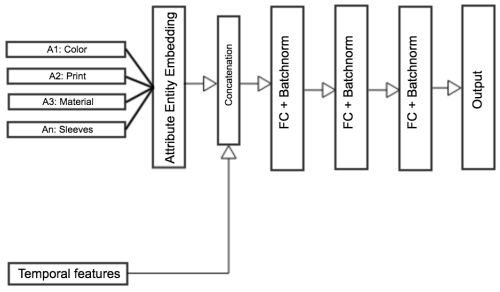
3 METHODOLOGY

While similar visual characteristics did not guarantee similar sales behaviour, our data does contain several similarly behaving time series having pricing, merchandising and visual factors in non-reducible ways. By that, we mean no intuitively explainable reduced representation for similarly behaving time series was able to be deduced. For example, conclusions like items in a particular price band and brand, or such combinatorially reducible groups behave similarly, cannot be made. The model we needed to build, thus, should learn to identify similarly behaving time series across latent parameters, and also take into account discounting, promotions, visibility variations in comparing the time series. A point in a time series is represented as

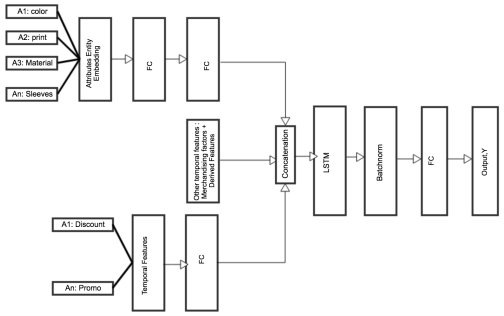
$$y_{it} = f(A_i, M_{i,t}, M_{i,t-1}, ..., M_{i,t-p}, D_{i,t}, D_{i,t-1}, ..., D_{i,t-p}) \quad (1)$$

where y_{it} is sales for item 'i' at time 't', A_i is attribute of the item 'i' like colour - blue, material - cotton etc., M_{it} indicate merchandising factors like discount, promotion for items 'i' at time 't', D_{it} are derived features like trend, seasonality which are inferred from data and affect the sales, p is number of time lag.

As mentioned in previous section, traditional time series models are not suitable choice for f. Hence, we work with machine learning models ranging from tree based models like Random Forest and various flavours of Gradient Boosted Trees, to deep learning models. We train two deep learning models, first of which uses Multi Layer Perceptron (MLP) architecture, and second is based on LSTM (chosen due to its ability to model long term temporal dependencies), to derive the relation f. Architectures of MLP and LSTM models are shown in Fig. 1.



(a) MLP Model



(b) LSTM Model

Figure 1: DNN Model Architectures

In the data, we see long tail behaviour that is typical characteristic of retail, with fewer items contributing to a majority of the sales. Due to this, we see variation of sales over several orders of magnitude. To address this high variance problem, we train our models at different scales - log and linear, and try a different set of loss functions. See Table 1 for more details.

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-linear relationship between target and regressors. Our scale is also large (~1 million styles or items listed at any point in time) that limits the utility of SVM-like models that do not scale well for large sets of data and hyperparameters.

Tree based models and MLP are trained in non-linear ARIMA [1] manner, where lagged values of time varying features are used

Table 1: Model Specification

Model	Criterion / Loss Function
Random Forest (RF)	Mean Squared Error (MSE)
Gradient Boosted Regression Trees (GBRT)	MSE and Huber
Light Gradient Machine (LGBM)	MSE and Poisson
CatBoost (CB)	MSE and Poisson
XGBoost (XGB)	MSE and Poisson
Attribute Embedding + MLP	MSE and Poisson
Attribute Embedding + LSTM	MSE and Poisson

to capture temporal dependencies. All the data and derived features are explained in the next section. We use lagged values of temporal features up to last 4 time steps ($p = 4$) . This was decided after some preliminary experiments and the intuition that temporal interactions over periods longer than 4 weeks are insignificant. Hyper-parameters of tree based models are optimized using Bayesian Hyper-parameter Optimization Technique [4]. We use documented best practices in deep learning along with some experiments and domain understanding to choose model hyper-parameters like learning rate. A value of 10^{-3} was found to be effective in most cases when used with cyclic learning rates [16]. We have observed an improved performance of the LSTM model when Dropout [9] and BatchNorm [15] are used. However, to avoid over-parameterization, we do not do very extensive neural architecture search, and use a simple network, shown in figure 1 [7]. Hyper-parameter optimization is done on the validation data.

LSTM model was trained in sequence to sequence [17] fashion using entire life-cycle data of a style, without explicitly coding temporal dependencies through lagged features as done with other models. We choose the LSTM approach, as several applications of this neural network architecture to sequences or time series [17] have shown promising results. Our aim was to experiment with the LSTM architecture to explore how well it learns non-linear temporal patterns in the data, especially in scenarios where reduced clusters in the attribute/design space are non representative of collective behaviour. We create 13 models for our study, as shown in Table [1]. Performance of models are assessed on test data.

3.1 Model Frameworks

Deep learning models are built using deep learning framework PyTorch [13] , and are trained on Azure instance containing 6 CPUs and a single GPU. Well know python packages are used for Tree based models, i.e. scikit-learn [14] is used for RF and GBRT; LGBM [10], CatBoost [6] and XGBoost [5] packages are used for other models.

4 EXPERIMENTS AND RESULTS

4.1 Data

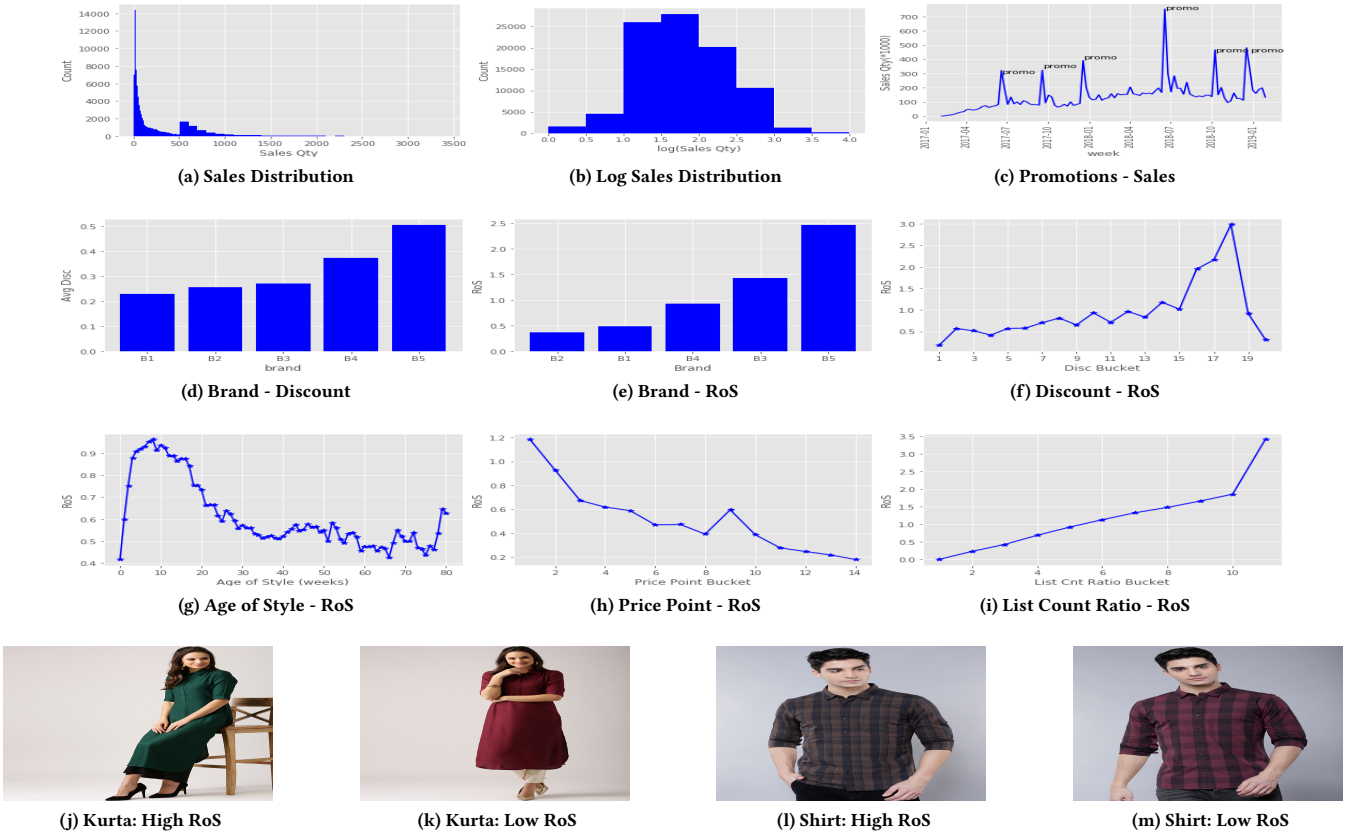
We use historical sales data of Myntra, a leading Indian fashion e-commerce company, to train our models. Experiments are conducted on data for 5 different article types. In our fashion ontology, an article type is a hierarchy level that contains items which can be characterised by a similar set of attributes, for example - Shirts, Casual Shoes, Tops, Kurtas etc. are article types, and particular items listed under these may be referred to as style or item alternately in our work. We use data for only those items which were catalogued or went live in the last two years. Data for items which went live in the first year are taken for training, and those which went live in next 6 months are used as validation set. The validation set is used to tune hyper-parameters of the models, using standard validation techniques. Finally, a test set of subsequent 6 months was used for measuring and reporting performance. The temporal length of time series for each style will vary, as they were listed for different duration. Minimum and maximum number of time series (TS), and their minimum and maximum length available across article type are provided in Table 2 to summarize our sequence lengths at play. Salient feature about the data and factors impacting sales are given in Figure 2.

Table 2: Time Series Details across Data

Data	Min No. of TS	Max No. of TS	Min TS Length	Max TS Length
Train	12,541	42,206	4	104
Valid	7,489	29,669	4	52
Test	6,732	18,364	4	26

We model promotions, discount, and list page views (visibility) along with fashion attributes of the style as external regressors. Some of these features are not known for future time steps at the time of prediction. Therefore we transform most of these features so that default values of promotions and discounts for future time steps can be easily approximated without remembering the training data. The details of engineered features are mentioned herein.

- **Fashion Factors:**
  - Fashion related *Attributes* such as colour, material etc. of a style are used. These attributes may be different for different article types. We embed [8] these attributes in order to both compress their representations while preserving salient features, as well as capture mutual similarities and differences. We learn these embeddings in the training phase. In our tree based approach, we use a simple one hot embedding [3] of attributes. Attribute values with frequency less than 1% are grouped into a dummy value to indicate values that may not be well represented in the data, as well as new and unseen values in future.



(a) Sales Distribution (b) Log Sales Distribution (c) Promotions - Sales

(d) Brand - Discount (e) Brand - RoS (f) Discount - RoS

(g) Age of Style - RoS (h) Price Point - RoS (i) List Count Ratio - RoS

(j) Kurta: High RoS (k) Kurta: Low RoS (l) Shirt: High RoS (m) Shirt: Low RoS

Figure 2: Salient Features of Data: Sales have Poisson Distribution in linear scale (a) and Normal Distribution in log scale (b). Promotions have positive impact on sales as observed by peak in (c). Not all brands are at same discount (d), hence not all brand will have same RoS (Rate of Sales - Ratio of Total Sales and Number of days for which style was live) (e). RoS increases with discount (f), whereas it first increases and then decreases with increase in Age of Style (g). RoS is higher for lower price point (h). Higher list count ratio leads to higher RoS (i). Effect of attribute on sales can be observed by comparing (j) with (k) and (l) with (m); (j) and (k) are Kurtas form same brand, at same price point and discount, though RoS of (j) is twice that of (k), due to color difference; similarly, (l) and (m) are shirts have same brand, price point and discount, but difference in color gives (l) a RoS which is 5 times of (m)

- **Merchandising Factors:**
  - *Discount*: In our initial analysis of the data, we found that most brands sold at an average (consistent) discount on our platform, while there were intra-brand variations in discounts that sometimes boosted sales on the retail platform. We capture the discount deviation from both the brand average, and overall retail platform average, hence, as we found this feature to contain more information than the item/style's absolute discount. A value of 0 in this case for future will mean that style will be sold at average brand/ platform discount. This feature also captured the non-linear and brand specific effects of discounting in fashion retail.
  - *Visibility*: Visibility features are derived from the list page views, which is the shelf space allocated to a style in an online store. List views ratio with respect to brand and platform are numerical measures of style visibility dispersion, and have big impact on observed sales, so we use them as features. List views given to a style depend on its sales, CTR, applied promotions etc. But in the absence of this information, usually platform average list views are given to new styles. Hence a value of 1 for future time steps is a reasonable assumption except for pre-decided special promotion days where the visibility can be appropriately boosted by a factor.
  - *Promotion*: To model the effect of sales drop just before and after a promotion, features like days to promotion and days from promotion are used. In Myntra and in the Indian retail scenario in general, certain country-wide observed holidays/occasions are promotional shopping festival days, such as Diwali, Valentine's day, etc. In the run up to a shopping festival (promotional), customers tend to postpone their buying till the promotional event, and immediately after a period of intense activity, we see a

significant lull in shopping enthusiasm. Hence the choice of maintaining a calendar like feature to indicate a count down to and from planned promotional events.

- **Derived Features:**
  - *Age of Style*: Shelf life of a style. With longer shelf life, the style's demand may decay with time.
  - *Trend and Seasonality*: To model a trend in interest over time, the number of weeks between experiment start date and the current date is used. In order to model seasonality in purchase patterns, first three terms of the Fourier transform of week of year are used as features. For a new item, these can be derived during prediction.
  - *Cannibalisation*: Cannibalisation is a commerce specific scenario where given that buyers/customers have a certain need, equivalent items may cannibalise each other's sales to meet that need. We create features like number of styles listed in a week, number of styles listed within the same brand in that week, number of styles listed by other brands in similar price ranges, etc. If all styles to be considered are available, along with their merchandising factors, these features can be inferred for new items; if not available then averages/medians may be used as representative values.

4.2 Testing and Evaluation

We use weighted mean absolute percentage error (wMAPE), equation 2, where the weight is the actual sales realised for an item.

$$wMAPE = \frac{\sum_{i=1}^{i=n} \sum_{t=1}^{t=t_i} |\hat{y}_{it} - y_{it}|}{\sum_{i=1}^{i=n} \sum_{t=1}^{t=t_i} y_{it}} \tag{2}$$

y_{it} and $\hat{y}_{it}$ is actual and forecasted sales of an item 'i' at time 't'. n is total number of items, t_i is the length of time series for item 'i'.

We choose to weight our MAPE by the item's actual sales in accordance with our tolerance for error in predicted values, so that the tolerance is lower with higher sales volumes. To illustrate the robustness of choice in wMAPE over MAPE, if actual sales for a set of items are 0, 5, and 10; and forecasted values are 1, 10 and 10; MAPE would be infinite, whereas wMAPE would be 0.4. In under-forecasting scenarios, errors are upper bounded by a wMAPE of 1; in overforecasting scenarios, wMAPEs may be arbitrarily high. We do not symmetrise our under-forecasting and over-forecasting scenarios, because over-forecasting leads to huge build up of inventory (due to need to order in lots or minimum order quantities). Generally speaking, the cost incurred per unit over-forecasted is much higher than the potential revenues missed per unit by under-forecasting. This is peculiar to retail supply chains where procurement lags are long (such as in fashion) and larger minimum order quantities apply.

When working with fashion buyers (known as planners) to operationalize our plans and evaluate our forecasts on real buys, we learnt that a relative priority of items is important to the procurement process since procurement happens in lots of minimum order quantity. An item with low forecasted sales may therefore not be ordered due to restrictions in buying budgets, time, and inventory holding capacity. Therefore, for an item that has higher actual sales realised relative to another, the forecasted sales should also be relatively higher so that ordering it ensures higher sell through rates as well as lesser inventory pile up. To capture this, we use the Pearson correlation, equation 3 and the Kendall tau, equation 4.

$$\rho_{y_i, \hat{y}_i} = \frac{E[y_i \hat{y}_i] - E[y_i]E[\hat{y}_i]}{\sqrt{E[y_i^2] - E[y_i]^2}\sqrt{E[\hat{y}_i^2] - E[\hat{y}_i]^2}} \tag{3}$$

$$tau = \frac{(P - Q)}{\sqrt{((P + Q + T) * (P + Q + U))}} \tag{4}$$

y_i and $\hat{y}_i$ are total actual and forecasted sales of item i. P is the number of concordant pairs, Q the number of discordant pairs, T the number of ties only in y_i, and U the number of ties only in $\hat{y}_i$.

Pearson Correlation ensures that forecasted values and actual values move together in the same direction, and Kendall Tau takes into account relative ordering of the quantities between forecasted and actual values.

For model tuning, we use Mean Squared Error (MSE) - equation 5, Poisson Loss - equation 6 and Huber Loss - equation 7.

$$MSE = \frac{\sum_{i=1}^{i=n} \sum_{t=1}^{t=t_i} (\hat{y}_{it} - y_{it})^2}{\sum_{i=1}^{i=n} t_i} \tag{5}$$

$$Poisson\ Loss = \sum_{i=1}^{i=n} \sum_{t=1}^{t=t_i} \hat{y}_{it} - y_{it} * log(\hat{y}_{it}) \tag{6}$$

$$Huber\ Loss = \frac{1}{2} \sum_{i=1}^{i=n} \sum_{t=1}^{t=t_i} \begin{cases} (\hat{y}_{it} - y_{it})^2 & \text{if } |\hat{y}_{it} - y_{it}| \leq \delta \\ \delta * |\hat{y}_{it} - y_{it}| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{7}$$

In a perfect world we would have preferred to optimize on one or all of the metric- wMAPE, PerasonR, or Kendall Tau, which we use to evaluate model, for model training. But none of these metric can be arrived from likelihood function, as is the case with MSE and Poisson. Under Gaussian assumption of target variable, likelihood function and MSE gives same solution, hence MSE is most preferred loss function for problem at hand. As, evident from figure 2(b) log transformed sales have Gaussian Distribution, hence MSE loss in log scale are used for model training. However, typically retail data such as ours 2(a) shows long tailed distribution in linear scale, hence we use Poisson loss in linear scale for learning model parameters. Huber Loss is used to minimize the effect of outlier on the training process. For each model, we specify the loss function before tabulating the wMAPE and ranking loss values.

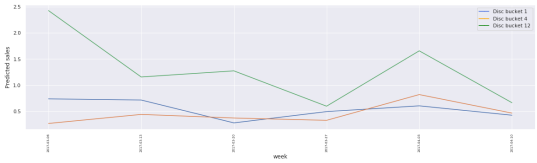
4.3 Results

Tables 3 and 4 show performance of top five models along with naive model, for two types of articles, namely shirts and casual shoes. For completeness, performance on other article types along with performance on training data is tabulated in Tables A1 to A8 of the Appendix. We observe that almost all ML based models outperform the naive average based projection model. XGBoost with MSE loss, when optimized in logarithmic scale gives best performance followed by GBRT. Among deep learning models, LSTM with Poisson loss, when optimized in linear scale gives best performance, MLP does not feature in top 5 performers, hence metric for it is not provided.

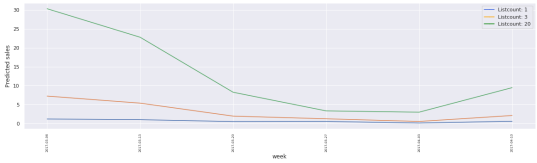
We provide example of good forecast - Fig. 4(a); Fig. 4(b) is an example where forecast is good for all but 3 weeks during which we

under-forecast. This is explainable as sales of this style just peaks after promotion period, whereas our model learns to forecast lower just after promotion, as general trend is; Fig. 4(c) is an example of bad forecast, we are heavily under-forecasting, this is being observed because this style is an exception in terms of sales for all styles belonging to its brand. These examples tells us that even though we have used tree based and deep learning models, results of which are considered to be not easily explainable - however, using derived features we can easily explain the results of our model.

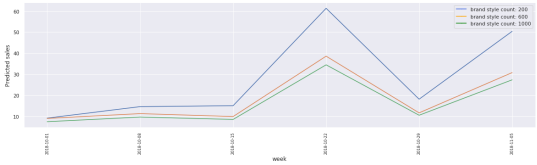
To illustrate the usefulness of transformed and derived features, we show forecast increases by increase in discount Fig. 3(a), higher discount bucket implies higher discount. Fig. 3(b) illustrates impact of increasing list count ratio on the forecast. As expected forecast increases with increase in list count ratio. Effect of cannibalization feature - brand style count, is shown in Fig. 3(c), increase in number of style from a brand decreases the forecasted sales, as would be expected.



(a) Discount - Forecast



(b) List Count Ratio - Forecast



(c) Brand Style Count - Forecast

Figure 3: Effect of derived features on forecast

4.4 Deployment in industrial setting

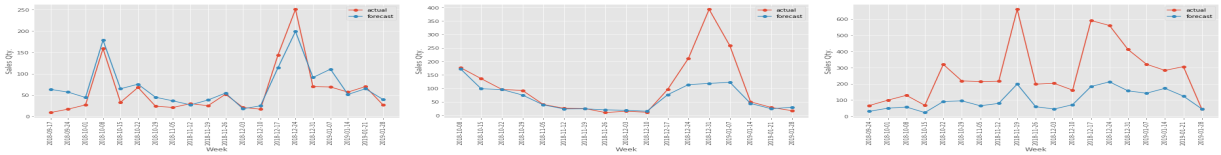
We have tested and deployed our models for the following fashion retail use cases at Myntra-Jabong. We also talk about futuristic scenarios where we are working to deploy our models.

- *Seasonal assortment Planning*: Fashion Retailers have to plan their assortment a year in advance due to manufacturing lead times. At the time planners do not have any information about the actual products, so they create all plans at attribute combination level and use an average based projection together with intuitive calls to allocate inventory budget. Our model when used with appropriate simulations can generate forecasts for all possible attribute combinations of styles. This result was used to decide the set of attribute combinations on which buying budgets should be spent, for two major footwear brands during the buying season of Autumn Winter (AW) 18. We saw a year-on-year improvement of 10% and 7% in the overall one month sell through rate of these brands for footwear category. Sell through rate is defined as the percentage of inventory at the season start which was sold during a specified period.
- *Product Selection in Roadshows* : Wherever a catalogue of items (with their descriptions and brands) is made available for a buyer to consume in events like Fashion Roadshows which buyers frequent to find out actually available assortment from different brands, our model can quickly compute projected sales for different items present in the roadshow. A buyer may use his/her intuition in addition to our model output to get directional information on which products to spend budget on. This deployment is a work in progress at our current organisation.
- *Drop Planning*: Purchase orders to manufacturers are placed long before the start of the season. However, to optimally utilize warehouse/ store space, deliveries / drop are taken in phased manner. Currently, all retailers plan drops at a fixed interval irrespective of how demand for an item is going to be. This leads to either lost sales or lot of inventory at hand. Our model's capability to provide good weekly sales forecast, evident from lower wMAPE at item-week level and figure 3(a), gives an opportunity to better plan drop by moving from manual to automated drop planning driven by data and machine learning. This use case is currently being tested at our organization

5 CONCLUSION AND FUTURE DIRECTIONS

We have presented the first large scale study for the demand forecast of new items in fashion. We have shown that careful feature engineering when used in conjunction with XGBoost, can be used to forecast demand for new items with reasonably good accuracies. While creating our models and features we have been cognizant of the fact that many features will not be available as is when forecasts are being generated for future period. Hence we have used innovative transformations so that we don't have to remember train data during forecast time, thereby reducing the computation and memory requirements during forecast generation. This has also allowed our models to be easily deploy-able for internet retailers where scale and performance are crucial deciding factors in operation. Section 4.4 lists business results achieved corresponding to the modeling outputs realised to demonstrate real world usefulness of our work.

Contrary to our initial expectations, DNN models (LSTM and MLP) did not show better performance over tree based models. LSTM seemed like a good choice of model theoretically since it has been shown to perform very well over various time series data, and is architecturally better suited to model long temporal dependencies. We intend to explore further in this direction by building an appropriate RNN architecture for demand forecasting that generalizes across datasets of different article types in fashion without



(a) Good Forecast (b) Under Forecast for some weeks (c) Bad Forecast

Figure 4: Actual vs. Forecasted: (a) wMAPE=0.34, is an example of good forecast, (b) wMAPE = 0.37, is an example of good forecast where we are under-forecasting for few weeks, (c) wMAPE = 0.63 is an example of bad forecast or heavy under-forecast

Table 3: Model Performance for Shirts on Test Data, 16,409 time series or items

Model	Criterion/Loss Function	wMAPE			PearsonR	Kendall Tau
		item-week	item	article type		
Naive (Avg. style-week sales) Model		0.97	0.82	0.39	0.26	0.43
XGBoost	MSE	**0.52**	**0.38**	**0**	**0.86**	**0.76**
GBRT	Huber	0.54	0.41	0.06	0.84	0.76
LSTM	Poisson	0.56	0.42	0.17	0.85	0.67
CatBoost	MSE	0.56	0.42	0.04	0.81	0.73
LGBM	MSE	0.57	0.43	0.03	0.81	0.74

Table 4: Model Performance for Casual Shoes on Test Data, 6,732 time series or items

Model	Criterion/Loss Function	wMAPE			PearsonR	Kendall Tau
		item-week	item	article type		
Naive (Avg. style-week sales) Model		1	0.85	0.43	0.36	0.46
XGBoost	MSE	**0.51**	**0.38**	0.11	0.89	**0.74**
GBRT	Huber	0.52	**0.38**	0.15	0.89	**0.74**
CatBoost	MSE	0.52	**0.38**	0.1	0.89	0.71
LGBM	MSE	0.54	0.39	0.05	0.88	0.71
XGBoost	Poisson	0.56	0.4	**0.04**	**0.91**	0.67

overfitting. We are also experimenting by including image based features in our forecasting models along with currently used textual attribute embeddings. Initial results seem encouraging with image based features, but we are still working on rigorous evaluation of these models on more datasets and finding scalable ways to run such models in real world scenarios.

REFERENCES

[1] [n. d.]. Autoregressive integrated moving average (ARIMA). https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average. Accessed: 2019-05-02.

[2] [n. d.]. H&M, a Fashion Giant, Has a Problem: $4.3 Billion in Unsold Clothes. https://www.nytimes.com/2018/03/27/business/hm-clothes-stock-sales.html. Accessed: 2019-05-02.

[3] [n. d.]. One Hot Encoding. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html. Accessed: 2019-05-02.

[4] James Bergstra, Daniel Yamins, and David Daniel Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. (2013).

[5] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[6] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. 2018. CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363* (2018).

[7] Valentin Flunkert, David Salinas, and Jan Gasthaus. 2017. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *arXiv preprint arXiv:1704.04110* (2017).

[8] Cheng Guo and Felix Berkhahn. 2016. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737* (2016).

[9] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).

[10] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *NIPS*.

[11] Ellen C. Mik. 2019. *New Product Demand Forecasting, A Literature Study.* Master's thesis. Vrije Universitat, Amsterdam. (In preparation).

[12] Maria Elena Nenni, Luca Giustiniano, and Luca Pirolo. 2013. Demand forecasting in the fashion industry: a review. *International Journal of Engineering Business Management* 5 (2013), 37.

[13] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine

Learning in Python . *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[15] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. 2018. How does batch normalization help optimization?. In *Advances in Neural Information Processing Systems*. 2483–2493.

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

[17] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[18] Sébastien Thomassey and Antonio Fiordaliso. 2006. A hybrid sales forecasting system based on clustering and decision trees. *Decision Support Systems* 42, 1 (2006), 408–421.

A APPENDIX

We list down results on some more article types for different types of models/loss functions used, and find that XGBoost with an MSE loss function consistently outperforms other choice of models and loss functions.

Table A1: Model Performance for Kurtas on Test Data, 9,161 time series or items

Model	Criterion/Loss Function	wMAPE			PearsonR	Kendall Tau
		item-week	item	article type		
Naive (Avg. style-week sales) Model		1.08	0.92	0.23	0.24	0.36
XGBoost	MSE	**0.6**	**0.46**	**0.08**	**0.85**	**0.71**
GBRT	Huber	0.64	0.49	**0.08**	0.83	0.69
LGBM	MSE	0.64	0.51	0.12	0.81	0.68
CatBoost	MSE	0.64	0.51	**0.08**	0.8	0.68
LSTM	Poisson	0.64	0.54	0.24	0.84	0.62

Table A2: Model Performance for Tops on Test Data, 10,618 time series or items

Model	Criterion/Loss Function	wMAPE			PearsonR	Kendall Tau
		item-week	item	article type		
Naive (Avg. style-week sales) Model		1	0.85	0.28	0.21	0.39
XGBoost	MSE	**0.51**	**0.37**	0.18	0.91	**0.75**
GBRT	Huber	0.54	0.39	0.2	0.91	0.73
XGBoost	Poisson	0.54	0.38	**0.06**	**0.92**	0.71
CatBoost	MSE	0.54	0.4	0.2	0.91	0.71
GBRT	MSE	0.55	0.39	0.15	0.9	0.73

Table A3: Model Performance for Tshirts on Test Data, 18,364 time series or items

Model	Criterion/Loss Function	wMAPE			PearsonR	Kendall Tau
		item-week	item	article type		
Naive (Avg. style-week sales) Model		1.01	0.86	0.27	0.3	0.4
XGBoost	MSE	**0.55**	**0.4**	0.04	**0.87**	**0.76**
CatBoost	MSE	0.57	0.42	**0.01**	0.84	0.72
LGBM	MSE	0.58	0.43	0.03	0.82	0.73
GBRT	Huber	0.59	0.44	0.06	0.84	0.74
LSTM	Poisson	0.6	0.46	0.07	0.85	0.64

Table A4: Model Performance for Shirts on Train Data, 31,581 time series or items

Model	Criterion/Loss Function	wMAPE			PearsonR	Kendall Tau
		item-week	item	article type		
Naive (Avg. style-week sales) Model		1.37	1.01	**0**	0.15	0.32
XGBoost	MSE	**0.3**	0.15	0.1	0.99	**0.89**
XGBoost	Poisson	**0.3**	**0.13**	**0**	**1**	0.87
GBRT	Huber	0.31	0.16	0.11	0.99	0.88
GBRT	MSE	0.34	0.17	0.1	0.99	0.87
LGBM	MSE	0.35	0.18	0.12	0.99	0.86

Table A5: Model Performance for Casual Shoes on Train Data, 12,541 time series or items

Model	Criterion/Loss Function	wMAPE			PearsonR	Kendall Tau
		item-week	item	article type		
Naive (Avg. style-week sales) Model		1.33	0.98	**0**	0.21	0.27
GBRT	Huber	**0.27**	**0.14**	0.1	**0.99**	**0.91**
LGBM	Poisson	0.32	0.13	**0**	**0.99**	0.86
GBRT	MSE	0.33	0.16	0.1	**0.99**	0.89
XGBoost	MSE	0.33	0.18	0.12	0.98	0.88
LGBM	MSE	0.36	0.18	0.12	0.98	0.86

Table A6: Model Performance for Kurtas on Train Data, 18,439 time series or ems

Model	Criterion/Loss Function	wMAPE			PearsonR	Kendall Tau
		item-week	item	article type		
Naive (Avg. style-week sales) Model		1.4	1.13	**0**	0.14	0.28
XGBoost	Poisson	**0.26**	**0.12**	**0**	**1**	0.86
GBRT	Huber	0.29	0.15	0.1	0.99	**0.88**
XGBoost	MSE	0.3	0.16	0.11	0.99	**0.88**
LGBM	Poisson	0.3	0.14	**0**	**1**	0.83
LSTM	Poisson	0.33	0.19	0.12	0.99	0.82

Table A7: Model Performance for Tops on Train Data, 23,801 time series or items

Model	Criterion/Loss Function	wMAPE			PearsonR	Kendall Tau
		item-week	item	article type		
Naive (Avg. style-week sales) Model		1.37	1.04	**0**	0.14	0.32
XGBoost	Poisson	**0.28**	**0.12**	**0**	**1**	0.86
XGBoost	MSE	0.29	0.15	0.1	0.99	**0.89**
GBRT	Huber	0.3	0.17	0.11	0.99	0.88
GBRT	MSE	0.34	0.17	0.1	0.99	0.87
LSTM	Poisson	0.36	0.13	0.04	**1**	0.82

Table A8: Model Performance for Tshirts on Train Data, 42,206 time series or items

Model	Criterion/Loss Function	wMAPE			PearsonR	Kendall Tau
		item-week	item	article type		
Naive (Avg. style-week sales) Model		1.37	0.95	**0**	0.18	0.3
XGBoost	Poisson	**0.31**	**0.15**	**0**	**0.99**	0.86
GBRT	Huber	0.32	0.17	0.11	**0.99**	0.88
XGBoost	MSE	0.32	0.17	0.11	**0.99**	**0.89**
GBRT	MSE	0.36	0.19	0.11	0.98	0.87
LGBM	MSE	0.37	0.2	0.12	0.98	0.86