TrendSpotter: Forecasting E-commerce Product Trends

Gayatri Ryali
igaryal@amazon.com
Amazon.com Inc.
Bengaluru, India

Shreyas S
shreyshs@amazon.com
Amazon.com Inc.
Bengaluru, India

Sivaramakrishnan Kaveri
kavers@amazon.com
Amazon.com Inc.
Bengaluru, India

Prakash Mandayam Comar
prakasc@amazon.com
Amazon.com Inc.
Bengaluru, India

ABSTRACT

Internet users actively search for trending products on various social media services like Instagram and YouTube which serve as popular hubs for discovering and exploring fashionable and popular items. It is imperative for e-commerce giants to have the capability to accurately identify, predict and subsequently showcase these trending products to the customers. E-commerce stores can effectively cater to the evolving demands of the customer base and enhance the overall shopping experience by offering recent and most sought-after products in a timely manner. In this work we propose a framework for predicting and surfacing trending products in e-commerce stores, the first of its kind to the best of our knowledge. We begin by defining what constitutes a trending product using sound statistical tests. We then introduce a machine learning-based early trend prediction system called *TrendSpotter* to help users identify upcoming product trends. *TrendSpotter* is a unique adaptation of the state-of-the-art InceptionTime model[6] that predicts the future popularity of a product based on its current customer engagement, such as clicks, purchases, and other relevant product attributes. The effectiveness of our approach is demonstrated through A/B tests, where we first showcase the effectiveness of our statistical test based labeling strategy, resulting in an incremental sales lift of 59 bps[1] across two experiments on home page and search page. Subsequently, we conduct a comparison between our machine learning model and the statistical labeling baseline and observe an additional sales gain of 14 bps, reflecting the importance of early identification of trending products.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Supervised learning**.

KEYWORDS

Trends, Time Series, Convolutional Neural Networks, E-commerce

[1]bps or basis points are a measure of percentages. 1 bps = 0.01%

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3615503

ACM Reference Format:
Gayatri Ryali, Shreyas S, Sivaramakrishnan Kaveri, and Prakash Mandayam Comar. 2023. TrendSpotter: Forecasting E-commerce Product Trends. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3583780.3615503

1 INTRODUCTION

In emerging markets like India, the rapid increase in use of smart devices and internet accessibility has resulted in widespread adoption of social media and e-commerce. Consequently, a significant number of internet users rely on influencers on services such as Instagram and YouTube to stay updated on latest trends and popular products. The insights and recommendations provided by influencers simplify the decision-making process for customers and save them their valuable time and effort. To serve the needs of such customers, we decided to simplify the discovery of trending products on e-commerce stores by providing the customers a single place to browse and buy trending products. In the current fast paced e-commerce environment, the trends are highly volatile and are prone to change week-in and week-out. To keep the customer up to date with the ever-changing trends, we choose to surface trending products on a weekly basis. Predicting and subsequently surfacing trending products to customers is a three stage problem. 1) Defining Trends - Since there is not a unique notion of trends on e-commerce stores, we first need to derive an appropriate and deterministic definition of trending products. 2) Predicting Trends - Using the said definition as labels we train ML models to predict the trendiness of various products for the future ahead. 3) Surfacing Trends - Using the customers' product search history, we surface similar and relevant trending products to them.

Defining trends is not straight forward given their exogenous nature. We find that various notions of trends have been proposed in the literature and often trends is defined as a growth in one or more domain relevant factors [2]. Given the e-commerce setting, we use growth in customer clicks as a prime indicator to determine the trendiness of a product. Inspired by [1], we prove that the week-on-week clicks growth of products follows a Cauchy distribution with its parameters as functions of past week clicks. We use the 90^{th} percentile of the said distribution as a threshold on the weekly growth, beyond which a product becomes trending. Due to the lack of quantitative metrics to determine the effectiveness of our definition, we resort to A/B testing. We observe that surfacing trending

products detected using our definition, resulted in 59 bps lift in sales. One significant disadvantage of identifying and surfacing previously trended products is the potential for displaying outdated trends because of the time delay in surfacing them. Since relevant trending products can only be surfaced after they have been detected, customers end up seeing event-relevant trends after their occurrence, thereby negatively affecting their shopping experience. To avoid this, we employ trend forecasting to ensure that customers stay updated on the latest ongoing fads.

Given the effectiveness and simplicity of our trends definition, we approach trend prediction as a time-series classification/regression problem, using the sound statistical definition as the ground-truth labels. Using the historical time-series data for each product, we can either estimate the probability of it trending in future as a classification problem or predict the future growth in click rate using regression. We advance the state-of-the-art model for time-series based prediction, InceptionTime [6], which we use to perform trend prediction. We call this as *TrendSpotter* and its architecture is shown in the figure 2. Our *TrendSpotter* model outperforms InceptionTime [6] and various other state-of-the-art time-series classification/regression models such as [17], [11].

By combining the power of statistics and ML, we enhance our ability to identify and track trends effectively. Our core contributions are summarised below.

(1) We present a set of statistical methods to identify trending products on e-commerce stores. We validate this definition of trending by performing A/B testing resulting in a combined revenue lift of 17 and 42 bps across home and search pages respectively.
(2) We cast the early identification of trends as a classification/regression problem. We train a custom and novel 1d-CNN based ML model, called the *TrendSpotter* using labels defined by the statistical model.
(3) We show quantitative and qualitative results that prove the efficacy of ML approach over statistical methods. Upon A/B testing, we observe that the ML based early trends show a lift of 14 bps and 13 bps respectively in sales and conversion over statistical detection approach.

In Section 2 we present the literature review and present our statistical methods to identify trends in time series data in Section 3. The proposed approach of predicting trends is presented in Section 4 followed by experimental analysis in Sections 5, 7. In Section 6, we discuss our NPMI based personalization strategy.

2 RELATED WORK

Trends - Exploratory studies [18] and [7], discuss the correlation between trends on Twitter and E-bay and the lag between them, with Twitter showing early trends. A plethora of methods found in the literature, model trends as statistical tests [9], probabilistic models [8] or via heuristics [15]. [8] primarily models/detects bursts, but fails to capture trends as it can classify a de-trending product as trending.The statistical tests [9] are sensitive to time window considered for detection. The heuristic models, though simple and flexible to define, have obvious drawbacks. For example [12], uses a rank based heuristic, where a product is trending if its popularity rank decreases to half its earlier value, however, this means a product can trend even if there is no change in given product's popularity but popularity of other products decrease. Particularly in industry, heuristic/statistical rule based methods have received a lot of attention like Google[14], Instagram[4]. All these methods can be used as trend detection models, i.e. they only detect products that were trending and they cannot predict a trend in advance. Moreover, there are no quantitative metrics to measure and compare these models [2].

Time-series prediction - Statistical time-series prediction methods such as AutoRegression, Moving Average and ARIMA[5] are too simplistic to capture intricacies of e-commerce time-series data. Due to recent success of deep learning techniques in fields of computer vision and NLP, similar methods have been proposed in the domain of time-series prediction. There are several 1d CNN-based state-of-the-art models such as InceptionTime [6], XceptionTime [11], OmniScale [13] and transformer based models such as TST [17], TS2Vec [16] and kernel based methods such as MiniRocket [3]. While these models perform better than the rest on certain time-series datasets with a palpable seasonal and/or trend components, they are not specifically designed and thus cannot cater to the e-commerce time-series data.

3 IDENTIFYING TRENDS

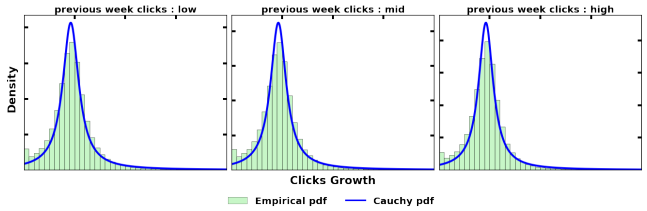
In this section, we design the trend detection heuristic based on clicks growth. We show that the ASIN clicks growth follows a Cauchy distribution and derive the necessary threshold on clicks growth to label an ASIN as trending.

Click growth versus other signals. In order to provide users with relevant trending products, the signal/s used to capture trends should represent the customer intent the best. We prefer growth in ASIN clicks as a signal over purchases and search impressions. Purchases are much sparser and are influenced by several factors other than customer intent such as prices, discounts and offers. Search impressions are noisy and are affected by factors such as promotions, advertisements etc. While clicks too can arise from several sources, in order to truly capture customer intent, we consider only those clicks that arise from user keyword searches.

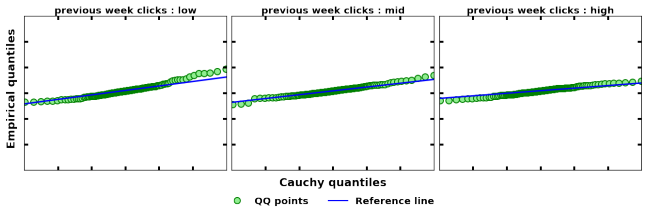
Let $c_w(a,t)$ be the number of clicks received by a product a over a window of length w, from $(t, t+w)$. Define $g_w(a,t) = \frac{c_w(a,t)}{c_w(a,t-w)}$ as the growth in product a's clicks from window $(t-w, t)$ to its subsequent window $(t, t+w)$. A product is said to be trending over a window $(t, t+w)$, if its growth, $g_w(a,t) \geq \tau_g$, where τ_g is the threshold on growth in clicks. Since, we aim to compute weekly trends, we observe growth in weekly clicks and window size is fixed to be 7 days unless mentioned otherwise.

Distribution of click growth. In figure 1, for various previous window clicks, k, we plot the distribution of clicks growth from the previous to the current window. As seen, the growth distribution varies with k and therefore, the growth threshold, τ_g is to be defined as a function of k.

Product a, that received $c_w(a, t-w) = k$ clicks over the window $(t-w, t)$, is said to be trending for the next window, $(t, t+w)$, if its clicks growth, $g_w(a,t)$ is greater than the 90^{th} percentile of all the growths of products with k clicks over the previous window. Let the $Q_g(k)$ be the 90^{th} percentile of clicks growth from window $w-1$ to w, for all products with $c_w(a, t-w) = k$. As seen in the figure 1, the clicks growth of all products with the previous window



(a) Distribution plots



(b) QQ plots

Figure 1: Observed vs Fitted distribution : Figure (a) comprises of the histogram of observed clicks growth data along with the corresponding fitted Cauchy distribution for 3 previous window clicks ranges - low, mid and high. As seen, the growth distribution varies with the previous window clicks. The goodness of fit for each previous clicks range is shown in figure (b) in the form of QQ plots. In each figure - (a),(b) both x,y axes are set to the same scale in the 3 plots.

clicks as k, approximately follows a Cauchy distribution with its distribution parameters as functions of k. The 90^{th} percentile of a Cauchy distribution with $x_0(k), \gamma(k)$ as the location and scale parameters respectively, is defined as $Q(k) = x_0(k) + \gamma(k)\tan[0.4\pi]$. We observe that for various previous week clicks, k, the 90^{th} percentile remains a constant, Q_g. We set the aforementioned growth threshold, τ_g to the computed 90^{th} growth percentile, Q_g.

*Definition 3.1 (**Trends**).* An product is said to be trending for a window $(t, t+7)$ if it satisfies the following conditions

- **C1.** Average number of clicks during the previous week should be greater than the average number of clicks during the past month i.e $\frac{c_7(a,t-7)}{7} \geq \frac{c_{30}(a,t-30)}{30}$
- **C2.** Number of clicks during the previous week must be above a set baseline, c_b in order to avoid substandard products i.e $c_7(a, t-7) \geq c_b$
- **C3.** growth in clicks from the previous week to the current week should at least be the computed growth threshold, $_g$ i.e $g_7(a, t) \geq \tau_g$

The products are likely to go out of stock or get suppressed, thus experiencing a drastic drop in clicks and resuming their original click behaviour upon re-stocking or unsuppression. To avoid consideration of products that went out of stock to coming back in-stock as trending, we introduce the condition **C1**. The condition **C2** helps eliminate substandard products from getting labelled trending, i.e. products whose clicks grow from 10 to 20, will not be considered trending even though the growth is 100%. The condition **C3** follows from the distribution of clicks growth.

Statistical model to validate the Trends definition. In the absence of reliable offline metrics to validate our proposed definition of trends, we resort to Weblab experiments by building a statistical model using the trends definition, 3.1. For a given week, $(t, t+7)$, the statistical model will surface products that are labelled trending over the past week, $(t-7, t)$ according to the definition 3.1. The weblab was successful with a revenue lift of 17, 42 bps on home and search pages, thus establishing the validity of using clicks growth as a trend signal and serving as a proof of concept that customers find trending products indeed engaging.

Drawbacks of Statistical Model. By construction, the statistical model surfaces products that were trending the previous week, during the current week. We find that these products are stale given the short-lived nature of trends. As seen in table 3, the heuristic model continues to surface event related products even after its occurrence thus showing stale trends to customers. Further, we found the overlap between the trending products that were shown during a week and the products that actually trended during the said week to be $\sim 10-12\%$ on an average. This indicates that the trends are rapidly changing and the heuristic method fails to capture/predict trends by its own definition. We overcome this by using an ML based learner to predict trends.

4 PREDICTING TRENDS

As the statistical model inherently can only detect trends post their occurrence, we rely on machine learning based models to forecast trends. The ML problem is to predict if the products that satisfy the conditions **C1** and **C2** in Definition 3.1 at a time t, will satisfy the condition **C3** on clicks growth over the unseen future window $(t, t+7)$. In this section, we discuss the architecture of *TrendSpotter* model that we built to predict trending products and its training details.

4.1 TrendSpotter

The *TrendSpotter* model predicts whether a given product is going to trend over the unseen window $(t, t+7)$ using the available historical information until the timestamp t. We have two variants of the *TrendSpotter* model, one to predict growth in clicks(regression) and other to predict probability of trending(classification). For both the variants, a product is labelled trending if its model output is greater than the appropriately set threshold. The model architecture, inspired by the state-of-the-art time-series classification model, InceptionTime [6], is shown in the figure 2. *TrendSpotter* model consists of the following two modules.

Trend-Inception module This module processes the historical time-series information available for each product. As depicted in the figure 3a, the input is passed parallely through 3 different 1d convolution layers, outputs of which are concatenated, batch normalized and average pooled before passing to the next module. The kernel sizes(1,3,7) were set to extract half-weekly and weekly aggregated features as the labels are based on weekly growth. We use average pooling instead of max pooling as we do not want to lose any aggregated information regarding the time-series data.

Future timestamp module At the time of prediction, we are aware of features such as month, date, day of the week etc. for the period of prediction. As shown in figure 3b, the module has

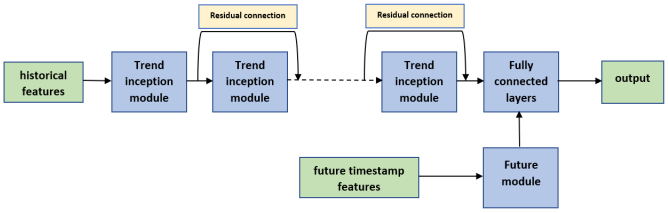
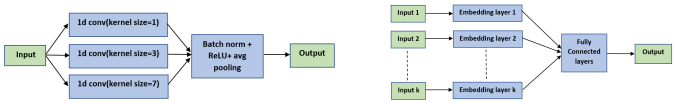


Figure 2: *TrendSpotter* model - the model consists of several trend-inception modules stacked one after another, to process the historical time-series data and a future module to process the future timestamp information. Outputs from both the models are concatenated and passed through fully connected layers in order to generate necessary output.



(a) trend-inception module **(b) future module**

Figure 3: *TrendSpotter* model components - Figure (a) depicts the architecture of trend-inception module, which is used to process historical information. Figure (b) depicts the future module, which is used to process categorical future time stamp features such as dates, days of the week, month for the prediction duration.

an embedding layer for each of the features used which are then concatenated and passed through fully connected layers.

Combining both modules As shown in the figure 2, the *TrendSpotter* model consists of several trend-inception modules stacked one after another, interlaced with residual connections. Output vectors from the last trend-inception module and the future module are concatenated and are passed through fully connected layers to obtain the necessary output i.e growth in clicks for the regression variant and probability of trending for the classification variant.

4.2 Model training

Features Features used can broadly be divided into 4 kinds - 1) Product signals such as historical clicks, purchases etc., 2) Statistical features like moving mean, standard deviation etc., of product signals, 3) Temporal features like day of the week, month etc., 4) Metadata like product category, title etc.

Training data We consider a database of ~800K products with 500 days of historical information of features mentioned above. At every timestamp, we filter out products that do not satisfy conditions **C1, C2** (3.1) resulting in the set of eligible products. It follows from our definition of trends that upto 10% of eligible products are labelled trending, making the dataset highly imbalanced. This is also seen in figure 1, where the growth distribution is heavily right skewed. Even though data is highly imbalanced, the individual class sizes are large enough. We sample ~ 1MM data points from the trending and non-trending classes to generate the training data. While the classification variant is trained on binary labels (1 for trending and 0 for non-trending), the regression one is trained on continuous labels (growth in product clicks).

Approach	Algorithm	Loss function	PRAUC
Regression	LSTM	L1	-
	InceptionTime	L1	+4.81
	XGBoost	L1	+5.7
	TST	L1	+6.17
	TrendSpotter	L1	+8.23
	TrendSpotter	L_q	+8.21
	TrendSpotter	**L1+L_q**	**+8.48**
Classification	LSTM	BCE	-
	InceptionTime	BCE	+6.12
	TST	BCE	+8.48
	XGBoost	BCE	+8.54
	TrendSpotter	BCE	**+8.75**

Table 1: Relative performance of *TrendSpotter* vs other models - Comparison of performance across various standard and SoTA ML algorithms against the trends-CNN model for both classification and regression streams. For all the models, we report relative metrics against LSTM. For the regression based models, we experiment with the loss functions - the standard L1 loss and the custom quantile loss, L_q. For the classification stream, we use BCE(Binary CrossEntropy) as the loss function. We highlight the best performing model for each stream.

Modified quantile loss function. We train the classification variant on binary cross entropy loss. For the regression variant, we can employ standard regression losses, like MSE loss or L1 loss. However, we design a novel loss function which imposes a higher penalty in the case of under-prediction for trending products and over-prediction for non-trending products. The reason being, for a trending product, under-predicting its growth might get it labelled non-trending. Similarly, for a non-trending product, over-predicting its growth might get it labelled trending. Let $\hat{g}_a, g_a$ be the predicted and the actual growth in clicks for a product a, whose trend label is $y_a \in 0, 1$. The modified quantile loss, L_q is defined as $L_q = \sum_a \max(0.9y_a(g_a - \hat{g}_a), 0) + \min(0.1y_a(g_a - \hat{g}_a), 0)$.

5 OFFLINE EXPERIMENTS

We train several models as mentioned in the table 1, on various combinations of features, hyper-parameters such as learning rates and model depth and choose the best performing combination for each model using a validation dataset. We use TSAI libray [10] to train SoTA models like TST[17], InceptionTime[6]. We measure the models' performance on recent test data, spanning over a year and comprising of 1MM+ data points. We report PRAUC(Area under the precision recall curve).

As seen in table 1, for both streams, the *TrendSpotter* model outperforms not only the standard but also the SoTA ML models including InceptionTime model [6], which it is based on. For the regression variant the best performance is seen when the *TrendSpotter* model is trained on both the loss functions - L1 and L_q (equation 4.2), thus proving the effectiveness of our custom loss function.

6 PERSONALIZATION

Surfacing personalized trending products based on the user's browse history positively impacts customer engagement. User information of any kind is anonymized, before we personalize trending products. For a given user, u, we define an interaction vector, $I_u \in \mathbf{R}^{|C|}$,

where C is the set of available product categories and |C| represents its cardinality. i^{th} element of the vector I_u is the probability that the user u interacts with corresponding product category c_i, which is defined as $I_u[i] = \frac{\log(1+n(u,c_i))}{\sum_{\hat{c}\in C}\log(1+n(u,\hat{c}))}$, where $n(u,c)$ is the number of clicks by a user u to the products from product category c. We use logarithm to attenuate the voluminous clicks on products in popular categories. We define trending categories, $\mathbf{C}_T$ as product categories with at least m trending ASINs. A customer might not always have interacted with a trending category, but might have with a product category similar to one or more trending categories. For example, a customer that highly interacted with products from 'SHIRT' category is more likely to click on products from 'PANTS' category as opposed to 'HOME DECOR'. Therefore it is necessary to compute the similarity between two categories.

In order to compute category-category affinity, we rely on NPMI - normalized pointwise mutual information. Let **U** denote the set of users. Let $P(U=u|C=c)$, $P(C=c|U=u)$ represent probabilities that a click to a product from category c arises from user u and vice versa respectively. Let $P(C=c)$ be the absolute probability of the category c and $P(C=c_2|C=c_1)$ be the transition probability from category c_1 to c_2.

$$\begin{aligned}
P(U=u|C=c) &= \frac{\log(1+n(u,c))}{\sum_{\hat{u}\in U}\log(1+n(\hat{u},c))} \\
P(C=c|U=u) &= \frac{\log(1+n(u,c))}{\sum_{\hat{c}\in C}\log(1+n(u,\hat{c}))} \\
P(C=c) &= \frac{\sum_{u\in U} n(u,c)}{\sum_{\hat{c}\in C}\sum_{u\in U} n(u,\hat{c})} \\
P(C=c_2|C=c_1) &= \sum_{u\in U} P(U=u|C=c_1)P(C=c_2|U=u) \\
P(C=c_1, C=c_2) &= P(C=c_1|C=c_2)P(C=c_2)
\end{aligned} \tag{1}$$

NPMI score between any two categories, c_1, c_2 is a number ranging from $[-1, 1]$. Higher the score, higher is the affinity between the two categories. We compute NPMI score as follows

$$NPMI(c_1,c_2) = \frac{\log P(C=c_1)\log(C=c_2)}{\log P(C=c_1, C=c_2)} - 1 \tag{2}$$

We define NPMI matrix, $N \in \mathrm{R}^{|C\times C_T|}$, with each element as the NPMI score between corresponding product category and trending category. For a user u, we define affinity vector $A_u \in \mathrm{R}^{|C_T|}$, each element of which denotes the affinity between the user u and corresponding trending category as $A_u = N^T I_u$. For each user u, using their affinity vector, A_u, we compute the top-K trending categories with highest affinity scores. Trending products from these categories are surfaced to the said user.

7 ONLINE A/B TESTS

To serve the best interest of our customers, we did a two stage A/B test. In the first stage we tested the effectiveness of statistical labels that help us generate data for training the ML model. In the second stage, we evaluated the ML predicted trends against the statistical labelling strategy. The weblab setting and results are summarised below. We track lift in business metrics such as sales, conversion(proportion of products clicks that led to purchases) and CTR(click-through rate) to quantify impact.

7.1 Statistical Trend Identification

We carried A/B tests to verify the soundness of the statistical trend identification where we surfaced the trending products for four weeks, on Home and Search pages to all customers. We added our trend widget in Treatment and not in Control, thus testing if showing the trending widget would fetch additional revenue compared to other recommendations shown in the Control group. Products to be shown on each page are personalized as follows. On the Home page, trending products are personalized according to the strategy mentioned in section 6 and for new users without much browse history, we showed products from most clicked trending categories. On search page, we showed trending products belonging to the category of current search query.

We observed a sales and conversion lift of 17 bps and 15 bps respectively on the home page. On the search page, a sales lift of 42 bps and conversion lift of 8 bps was seen. While on both the pages the trending product recommendations were deemed useful by the customers, the search widget significantly outperformed the home widget. This can be attributed to the the strong personalization signal available on search page (current search query category) as opposed to the home page (previous product clicks).

7.2 ML Trend Prediction

In order to compare the ML model performance against the statistical model, we ran an A/B test with treatment/s as the ML model/s and the statistical model as control. We chose the two best performing models - XGBoost model(as a baseline) and our *TrendSpotter* model as the two treatments, T1 and T2 respectively. In late October, we initiated a 28-day A/B test in the Indian marketplace, which yielded positive results for both ML treatments as summarized in table 2. Notably, the A/B testing coincided with two significant events, Halloween and Children's Day, providing valuable real time data for comparing the performance of the ML models against the statistical model.

As seen in table 2, both treatments showed an increase in conversion and sales. We observed an increment of $\sim$ 250 bps in mean click-through rate for prediction models with respect to the statistical model. Table 2 shows that even among the major product cohorts like Fashion, Wireless and Consumables, the ML based treatments outperformed control resulting in higher revenue across such groups. We observe that our *TrendSpotter* model (T2) had nearly 100 times the sales and twice the conversion rate compared to the baseline XGBoost model(T1) with respect to the control. This indicates the sheer superiority of the quality of trends provided by our *TrendSpotter* model. These observed metrics can be explained by following the artefacts noticed in A/B test data

- **Stale trends in control**: Both T1 and T2 surfaced Halloween related products only till Oct-31. However, the control showed such products, well past Halloween(till around Nov-7).
- **Early trends in treatments**: Both T1 and T2 picked up products relevant for children's day (Nov 14) by Nov-7 whereas the control picked the same much closer to the event date(Nov-10) and continued to show them well past Nov-20.
- **Diversity**: November being season of weddings in India, all the treatments picked products related to this event, but

Metric	Cohort	Baseline(T1)	*TrendSpotter*(T2)
Sales	All	0 bps	**14 bps**
	Fashion	8 bps	**10 bps**
	Wireless	25 bps	**85 bps**
	Consumables	**47 bps**	40 bps
Conversion	All	8 bps	**14 bps**
	Fashion	**18 bps**	17 bps
	Wireless	12 bps	**56 bps**
	Consumables	19 bps	**41 bps**

Table 2: A/B test results of ML prediction vs statistical identification - the table consists of performance of both the treatments with respect to control in terms of various business metrics over different product cohorts. As seen both the treatments consistently outperform the control and *TrendSpotter* outperforms the baseline XGBoost model by a huge margin.

(a) trending categories related to winter, captured by *TrendSpotter*

(b) trending categories related to Indian weddings, captured by *TrendSpotter*

Figure 4: Categorical diversity of *TrendSpotter* : *TrendSpotter* captures an eclectic set of trending products spread across all the theme-relevant categories. For winter, we have products that belong to male and female jackets, mufflers, gloves, blankets, heaters and beauty products. For the wedding season, we see products ranging from men and women's ethnic wear to jewellery, bangles, fancy clutches and hair accessories to ceremony props. Such a diversity is not seen in case of the heuristic model, where the products belong to a few categories like men and women's clothes.

T1 and T2 had more a diverse set than the heuristic model. Similar behaviour was seen in case of winter related products as November is the onset of winter season, as seen in figure 4

- **Sensitivity**: The ML models are sensitive to faint signals that indicate rather niche trends. This we observed when ML based treatments surfaced products like lamps, oil, incense sticks well before a local lamp festival. The heuristic model completely missed the same as the thresholds on observed clicks were sensitised to identify most popular trending products.

We observe that these traits are seen in the quality of the trending products surfaced by these methods even beyond the A/B test duration. For various events, we analyse the relevant trending products by the prediction and heuristic models to compute their coverage a week before and a week after the experiment. The coverage is measured as the number of event-relevant products and their

(a) trending products related to FIFA

(b) trending products related to F1

Figure 5: Niche trends captured by *TrendSpotter*: FIFA and F1 are relatively niche in India. On the left are a few of the trending products related to the FIFA trend like jerseys and football shoes surfaced during the FIFA 2022 world cup season. On the right, are a few of the F1 teams' merchandise like Mercedes cap and Ferrari shoes that were surfaced since the start of F1 2023 season.

categories. As seen in table 3, the prediction model(Treatments) captures relevant, early and diverse trends in comparison to the stale ones captured by the heuristic model(Control). Additionally, we see that the prediction model is successful in surfacing trending products relevant to niche trends such as FIFA and F1(both of which are less popular in India) in the figure 5. The heuristic model fails to capture such data.

Theme/Event	Early Trends		Stale Trends	
	Products	Categories	Products	Categories
Halloween	+10.0%	+33.3%	-92.3%	-33.3%
Children's Day	+230.7%	+200.0%	-88.9%	0.0%
Christmas	+7.3%	+60.0%	-90.4%	-75.0%
Valentine's Day	+103.9%	+150.0%	-95.04%	-75.0%

Table 3: Relative coverage of *TrendSpotter* with respect to Heuristic model - We look at the percentage of increment/decrement in the number of trending products and categories generated by *TrendSpotter* with respect to Heuristic model, a week before and after an event, for early and stale trends respectively. As seen, our *TrendSpotter* model has much higher coverage before the event and much lower coverage post the event, compared to the Heuristic model. As it is desirable to surface trending products in anticipation of the future event, we infer that the *TrendSpotter* model surfaces timely, relevant and diverse set of trending products as opposed to their heuristic counterpart.

8 CONCLUSION

In this paper, we present the derivation and the necessary validation (using a successful A/B testing) of the definition of trending products on e-commerce stores. Using the definition as labels, we pose trend prediction as an ML problem. We build the *TrendSpotter* model, which captures the intricacies in the product history to determine its future trendiness. Due to the relevant, timely and diverse trends surfaced by our *TrendSpotter* model, it experienced a monumental gain of 14 bps in revenue and 13 bps in paid units, over its heuristic counterpart, there by leading to its launch on the home page.

REFERENCES

[1] Justin Cheng, Lada Adamic, P. Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. 2014. Can Cascades Be Predicted?. In *Proceedings of the 23rd International Conference on World Wide Web* (Seoul, Korea) *(WWW '14)*. Association for Computing Machinery, New York, NY, USA, 925–936. https://doi.org/10.1145/2566486.2567997

[2] Justin Cheng, Lada A. Adamic, Jon M. Kleinberg, and Jure Leskovec. 2016. Do Cascades Recur?. In *Proceedings of the 25th International Conference on World Wide Web* (Montréal, Québec, Canada) *(WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 671–681. https://doi.org/10.1145/2872427.2882993

[3] Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. 2021. MiniRocket: A Very Fast (Almost) Deterministic Transform for Time Series Classification. (2021), 248–257.

[4] Instagram Engineering. 2015. *Trending on Instagram.* https://instagram-engineering.com/trending-on-instagram-b749450e6d93

[5] Rob J. Hyndman and Yeasmin Khandakar. 2008. Automatic Time Series Forecasting: The forecast Package for R. *Journal of Statistical Software* 27, 3 (2008), 1–22. https://doi.org/10.18637/jss.v027.i03

[6] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. InceptionTime: Finding AlexNet for Time Series Classification. *Data Mining and Knowledge Discovery* (2020). https://doi.org/10.48550/arXiv.1909.04939

[7] Sanjay Kairam, Meredith Ringel Morris, Jaime Teevan, Dan Liebling, and Susan Dumais. 2013. Towards Supporting Search over Trending Events with Social Media. In *Proceedings of ICWSM 2013* (proceedings of icwsm 2013 ed.). AAAI - Association for the Advancement of Artificial Intelligence. https://www.microsoft.com/en-us/research/publication/towards-supporting-search-over-trending-events-with-social-media/ Best Paper Honorable Mention.

[8] Jon Kleinberg. 2002. Bursty and Hierarchical Structure in Streams. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Edmonton, Alberta, Canada) *(KDD '02)*. Association for Computing Machinery, New York, NY, USA, 91–101. https://doi.org/10.1145/775047.775061

[9] Henry B. Mann. 1945. Nonparametric Tests Against Trend. *Econometrica* 13, 3 (1945), 245–259. http://www.jstor.org/stable/1907187

[10] Ignacio Oguiza. 2022. tsai - A state-of-the-art deep learning library for time series and sequential data. Github. https://github.com/timeseriesAI/tsai

[11] Elahe Rahimian, Soheil Zabihi, Seyed Farokh Atashzar, A. Asif, and Arash Mohammadi. 2019. XceptionTime: A Novel Deep Architecture based on Depthwise Separable Convolutions for Hand Gesture Classification. *ArXiv* abs/1911.03803 (2019).

[12] Bidisha Samanta, Abir De, Abhijnan Chakraborty, and Niloy Ganguly. 2017. LMPP: A Large Margin Point Process Combining Reinforcement and Competition for Modeling Hashtag Popularity. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2679–2685. https://doi.org/10.24963/ijcai.2017/373

[13] Wensi Tang, Guodong Long, Lu Liu, Tianyi Zhou, Jing Jiang, and Michael Blumenstein. 2020. Rethinking 1D-CNN for Time Series Classification: A Stronger Baseline. *ArXiv* abs/2002.10061 (2020).

[14] Google Trends. [n. d.]. *Get started with Google Trends.* https://support.google.com/trends/answer/6248105?hl=en&ref_topic=6248052

[15] Peter R. Winters. 1960. Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science* 6, 3 (1960), 324–342. http://www.jstor.org/stable/2627346

[16] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yu Tong, and Bixiong Xu. 2021. TS2Vec: Towards Universal Representation of Time Series.

[17] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. A Transformer-Based Framework for Multivariate Time Series Representation Learning *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 2114–2124. https://doi.org/10.1145/3447548.3467401

[18] Haipeng Zhang, Nish Parikh, Gyanit Singh, and Neel Sundaresan. 2013. Chelsea Won, and You Bought a t-Shirt: Characterizing the Interplay between Twitter and e-Commerce. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (Niagara, Ontario, Canada) *(ASONAM '13)*. Association for Computing Machinery, New York, NY, USA, 829–836. https://doi.org/10.1145/2492517.2500302