# PRODIGY: Product Design Guidance at Scale

Sambeet Tiady
Amazon
tiadys@amazon.com

Anirban Majumder
Amazon
majumda@amazon.com

Deepak Gupta
Amazon
dgupt@amazon.com

## ABSTRACT

Growth of e-commerce has enabled the creation of thousands of small-scale brands. However, these brands lack information on a) what new products to develop and b) how to refine existing products to improve business metrics. We present a comprehensive **Pro**duct **D**esign **I**nsights and **G**uidance service (named PRODIGY) that mines product attributes data available on e-commerce platforms and surface insights on a) new product development and b) product refinement. Our core contribution is a novel demand forecasting model for product designs based on a notable extension of the recently proposed FTTransformer [3] architecture combined with a self-supervised pre-training task, akin to Masked Language Modeling (MLM [2]). For the product refinement use-case, we present a novel algorithm by embedding the design search in a data-density approximator, namely Conditional Variational Autoencoder. We run a thorough and comprehensive set of experiments and establish that PRODIGY achieves significant improvement in demand prediction as compared to state-of-the-art alternatives. Finally, we present our findings from an online experiment where PRODIGY helped to launch new products with +20% lift in sales and +1.3% lift in product ratings.

## CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Neural networks**; **Supervised learning by regression**; *Multi-task learning*; • **Information systems** → *Data mining*; *Online shopping*; • **Applied computing** → *Online shopping*;

## KEYWORDS

Product design insights and guidance, Transformers for tabular data, Demand forecasting

**ACM Reference Format:**
Sambeet Tiady, Anirban Majumder, and Deepak Gupta. 2023. PRODIGY: Product Design Guidance at Scale. 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/10.1145/3583780.3615494

## 1 INTRODUCTION

The rapid proliferation of e-commerce sites, has fueled the growth of many small-scale emerging brands. These brands often lack intelligence and R&D investment on what product features they should focus on to win and retain customers. For example, a brand launching a new pressure cooker, may be unaware that anodized aluminum proffers better heat conduction as compared to stainless steel. Further, they may lack intelligence on the cost-benefit impact, i.e. how much additional revenue is the fix expected to generate in future. E-commerce websites have an abundance of product-related information available in their catalog, ranging from product specifications, performance metrics to product reviews and ratings. This rich data can be mined to generate product intelligence reports to brands via a web interface. Specifically, one can address the following questions: (1) **new product development:** *for a product category, what are the most important features to focus on and what is the market opportunity?*, (2) **portfolio enhancement:** *for existing products, what are the minimal changes to incorporate to make it perform better?* (Refer Figure 1 for a user flow of both cases). These insights will help brands to design better products, as well as drive more traffic and revenue to the e-commerce website.

*CIKM '23, October 21–25, 2023, Birmingham, United Kingdom*
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3615494

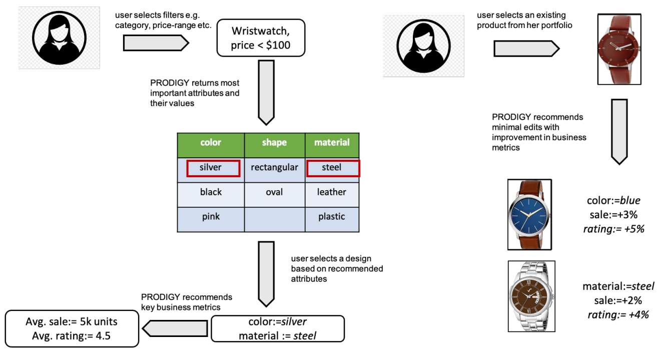
user selects filters e.g. category, price-range etc.
Wristwatch, price < $100
PRODIGY returns most important attributes and their values
color
shape
material
silver
rectangular
steel
black
oval
leather
pink
plastic
user selects a design based on recommended attributes
PRODIGY recommends key business metrics
Avg. sale:= 5k units
Avg. rating:= 4.5
color:=*silver*
material := *steel*
user selects an existing product from her portfolio
PRODIGY recommends minimal edits with improvement in business metrics
color:=*blue*
sale:=+3%
*rating:= +5%*
material:=*steel*
sale:=+2%
*rating:= +4%*

**Figure 1: Sample guidance offered by PRODIGY for new product development and portfolio enhancement workflows.**

To address these questions, we present a web-based product guidance service named PRODIGY (**Pro**duct **D**esign **I**nsights and **G**uidance) that mines product intelligence from data available on e-commerce websites across all product categories. PRODIGY helps sellers and brands to increase their product portfolio on the e-commerce site and recommends actions to improve the quality of their existing products. The core to our system is a demand prediction model that estimates key business metrics given a product design as the input. Product design refers to product specifications or attributes, which are represented as key-values pairs specific to a product category, e.g., attribute key is *material* whose legitimate values are "steel", "aluminum", etc.

Unlike traditional forecasting, where the goal is to predict future demand using historical data, forecasting using product design requires the demand estimate at the design phase without any historical data. Additionally, design specifications can be incomplete - designers may want an initial estimate with some high-level attributes before diving into finer details. For example, a brand could explore market opportunity for pressure cookers with *material:=steel* and *volume:=5 liters*, when other parameters like thickness or color are undefined. This requires our forecasting model to be resilient to missing values in the product specifications.

For the new product development use-case, we run model explainability techniques on the demand forecast model to generate insights on attribute keys and values, which are surfaced to brands to educate them about new categories. For the portfolio enhancement use-case, we consider recommending *minimal edits* of existing products which will generate higher sales, with the constraint that, attribute combinations recommended by our algorithm are feasible and not an outlier among products sold on the e-commerce website. For example, if we recommend changing the material from "stainless steel" to "anodized aluminum", we also need to increase the base thickness compared to stainless steel cookers. This requires us to efficiently search for a counterfactual [1] product design, which we achieve via a data-density approximator (Conditional Variational Auto-Encoder or CVAE) for feasible designs. We present a novel algorithm to recommend minimal edits to existing products with quantified impact, while ensuring that the final product does not fall off the data manifold. Specifically, we make the following contributions in this paper:

(1) We present a novel algorithm based on the state-of-the-art FTTransformer [3] model for tabular data-sets. We enhance the model with an inter-sample attention layer and a pre-training step which significantly improves its performance.
(2) We present a novel multi-task architecture to jointly estimate demand and data density of product designs. The data-density approximator helps in counterfactual design search, while the demand estimation task ensures the recommendations lead to improvement in business metric.
(3) For scalability across all product categories, we apply a clustering algorithm on the attribute space and train a model for each cluster. Experimental evidence suggests, this strategy reduces the operational load of maintaining multiple models by 70% and improves forecast quality by up to 15%.
(4) We run a comprehensive set of experiments that suggests that our algorithm generates more accurate forecast and better quality of product insights compared to other baselines. We perform extensive ablation studies to gauge the importance of each of our components.
(5) PRODIGY was deployed to production to recommend product quality enhancements to sellers in an emerging marketplace. It was observed in an A/B test, that products launched through PRODIGY recommendations drive 40% higher page views, 20% higher sales and 1.3% improvement in product rating compared to incumbent techniques.

## 2 PRODUCT DESIGN GUIDANCE

In this section, we describe the technical components of PRODIGYȦt a high-level, our approach is the following: we train a forecast model that estimates the demand given a product design as input, possibly with incomplete specifications. We then apply explainability techniques to generate useful insights on product attributes. Finally, we leverage the conditional data-density approximator (CVAE) and use it in a lock-step fashion with the forecast model to help brands synthesize minimal edits of existing products with improved performance. First, we introduce some notation.

**Notations** Let $\mathcal{A}_i$ denote the set of values for attribute $i$. A product design is represented as a $S$-dimensional vector $\boldsymbol{x} \in (\mathcal{A}_1 \cup \{\emptyset\}) \times (\mathcal{A}_2 \cup \{\emptyset\}) \times \cdots (\mathcal{A}_S \cup \{\emptyset\})$. Note that some components of $\boldsymbol{x}$ could be missing owing to incomplete specification of the design. Throughout the paper, we use boldface uppercase letter to indicate tensors of higher dimensions ($> 1$), boldface lowercase letter to indicate vectors and lower case letters to indicate scalars. We use $F$ to denote embedding dimension and $B$ as batch size. Our solution leverages an encoder-decoder architecture. We use the notation $\boldsymbol{E}^{(i)}$ (or $\boldsymbol{D}^{(i)}$) to denote the output from the $i^{th}$ layer of the encoder (decoder respectively).

### 2.1 Demand Prediction

Our goal is to train a model $y = f(\boldsymbol{x})$ that takes the (possibly incomplete) design $\boldsymbol{x}$ as input and predicts its demand $y$, i.e. the number of units the product will be able to sell over a fixed time horizon (say, 6 months) if it were launched. Gradient-boosted trees such as XgBoost [1] or LightGBM [7] are commonly preferred on tabular data-sets due to their superior performance. However, their performance was observed to be below par for our solution. We present a novel transformer-based architecture to generate this demand forecast. Our work is based on the recently proposed FTTransformer [3] model, which we will briefly describe now, followed by specific enhancements catered to our use-case.

The FTTransformer model learns a transformation of both numeric and categorical features via an embedding layer. For a numeric feature $x_i$, it applies a transformation $\boldsymbol{e_i} = \boldsymbol{\alpha_i} \cdot x_i + \boldsymbol{b_i}$ where $\boldsymbol{\alpha_i}, \boldsymbol{b_i} \in \mathbb{R}^F$ are model parameters that maps the scalar input $x_i$ to an $F$-dimensional vector $\boldsymbol{e_i}$ through element-wise multiplication. On the other hand, for a categorical feature $x_j$, it does a lookup on the embedding table $\boldsymbol{B} \in \mathbb{R}^{|\mathcal{A}_j| \times F}$ i.e. $\boldsymbol{e_j} = \boldsymbol{b_j} + \mathbb{1}^T \cdot \boldsymbol{B}$, where $\mathbb{1}^T$ is an one-hot encoded representation of the categorical feature, and $\boldsymbol{b_j} \in \mathbb{R}^F$ is a model parameter. Finally, the embeddings of each of the $S$ features are stacked up to generate a representation $\boldsymbol{E}^{(0)} \in \mathbb{R}^{S \times F}$ of the input $\boldsymbol{x}$, which is passed through $i$ transformer layers to get the output $\boldsymbol{E}^{(i+1)}$.

We make a few enhancements to the vanilla FTTransformer model (c.f. Figure 2 (left)). First, we add a single inter-sample attention layer [16] where the attention is computed across different samples in a mini-batch. Specifically, given input $\boldsymbol{E} \in \mathbb{R}^{S \times F}$, we compute its similarity with all other samples $\boldsymbol{E'}$ in the mini-batch, in terms of dot product. Then, the attended form of $\boldsymbol{E}$ is computed as the weighted average of all samples in the mini-batch. We apply a self-attention layer [8] on the output of FTTransformer to obtain a latent representation $H \in \mathbb{R}^{B \times F}$. Finally, we pass the representation $H$ through a stack of feed-forward layers to predict the target.

### 2.2 Handling Incomplete Designs

To allow incomplete specification of designs, the forecast model needs to be robust with respect to missing values in input. We leverage a pre-training step for the forecast model, akin to self-supervised masked language modeling (MLM [2]) task, where we drop attribute values randomly from fully specified designs and

[1]we use the term *counterfactual* in the sense that these new products do not currently exist on the e-commerce site.

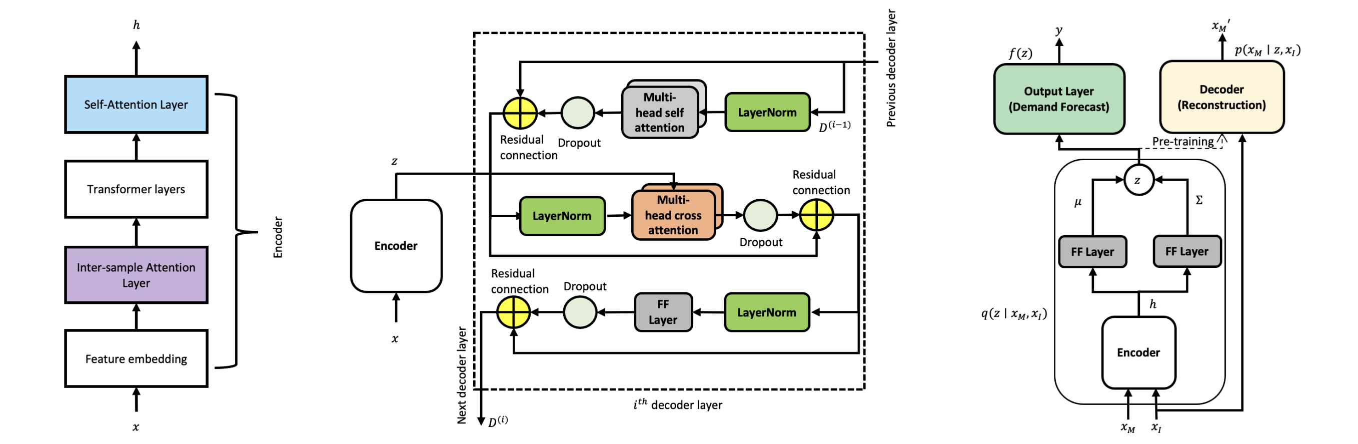

**Figure 2: On left, we show the PRODIGY encoder, while the decoder is depicted at the center. On right, we present the high-level architecture, with a shared encoder and separate layers for forecasting and data-density estimation (best viewed in color).**

require the model to predict their values. We replace known values of attribute $j$ (selected randomly) with a special token $< \text{MASK}_j >$ and the model is trained to reconstruct its value. In other words, we teach the model to *guess* the value of the missing attribute based on known values of other attributes. This is achieved with a decoder, which we describe in detail in Section 2.4. Note that unlike in traditional MLM task, we leverage different mask tokens for each attribute. The embeddings for the special tokens $\{< \text{MASK}_j >\}_{j=1}^{S}$ are learned during the pre-training step. Any unspecified design attribute during the model training or evaluation step is replaced with the corresponding $< \text{MASK} >$ token prior to doing a forward pass through the model.

## 2.3 Product Guidance Insights

Post model training, we leverage explainability techniques namely, Shapley values [9] which assign an attribution score to each attribute. The attribution scores are aggregated across all products within a category to generate attribute ranking. The aggregated scores are used to surface various insights for the new product development use-case: (1) what are the critical attributes for a given category (say, "pressure cooker"), (2) what are the most important values for a given attribute (say "color")?

## 2.4 Finding Minimal Edits

The goal is to find minimal attribute changes to a product to make it perform better. One simple approach is to take a gradient step in the $H$-space, in the direction of increasing sales and decode the resulting product. We achieve this via the following transformations:

$$\begin{aligned} \boldsymbol{h} &\leftarrow \text{Encoder}(\boldsymbol{x}) \\ \boldsymbol{h}' &\leftarrow \boldsymbol{h} + \alpha \cdot \left(\frac{\partial y}{\partial H}\right)_{H=\boldsymbol{h}} \\ \boldsymbol{x}' &\leftarrow \text{Decoder}(\boldsymbol{h}') \end{aligned} \tag{1}$$

Keeping the learning rate $\alpha$ small enough ensures that $\boldsymbol{x}'$ is a minimal edit of $\boldsymbol{x}$. However, gradient ascent may not respect correlations among attributes. The key requirements here are: (a) the new design should be an improvement, and (b) it is representative of products typically sold on the e-commerce website. In order to meet the second condition, we require a data-density approximator that estimates $p(\boldsymbol{x}')$. If $p(\boldsymbol{x}') \ll p(\boldsymbol{x})$, design $\boldsymbol{x}'$ is likely to be an outlier. We split the feature vector $\boldsymbol{x}$ into two disjoint sets: *immutable* $(\boldsymbol{x}_I)$ and *mutable* $(\boldsymbol{x}_M)$ parameters. The splits are user-specified, e.g. the user may want to condition on a specific brand, i.e. $\boldsymbol{x}_I = \{\text{brand}\}$.

We incorporate a CVAE [15] component in our model, to estimate $p(\boldsymbol{x}_M \mid \boldsymbol{x}_I)$. It consists of a decoder $p(\boldsymbol{x}_M \mid \boldsymbol{z}, \boldsymbol{x}_I)$ and a variational encoder $q(\boldsymbol{z} \mid \boldsymbol{x}_I, \boldsymbol{x}_M)$ which is used to approximate the true posterior $p(\boldsymbol{z} \mid \boldsymbol{x}_M, \boldsymbol{x}_I)$. The encoder takes $\boldsymbol{x} = (\boldsymbol{x}_M, \boldsymbol{x}_I)$ as input and emits $\boldsymbol{h} \in \mathbb{R}^{B \times F}$. The variable $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ is a stochastic node in the computation graph whose mean $(\boldsymbol{\mu})$ and variance $(\Sigma)$ are estimated through 2-layer feed-forward layers given $\boldsymbol{h}$ as input. Figure 2 (center) shows the components of a single decoder layer. The $i^{\text{th}}$ decoder layer receives the input $\boldsymbol{D}^{(i-1)}$ with the input $\boldsymbol{D}^{(0)}$ being sourced from the encoder (i.e. $D^{(0)} = \boldsymbol{Z}$). The cross attention layer computes attention scores between encoder output $\boldsymbol{z}$ and decoder input $\boldsymbol{D}^{(i)}$. The decoder consists of a stack of five such layers and estimates $p(\boldsymbol{x}_M \mid \boldsymbol{z}, \boldsymbol{x}_I)$.

PRODIGY leverages a novel multi-task architecture to jointly learn (a) demand forecast and (b) data density for new designs. We leverage a shared encoder and separate decoder for each task (refer to Figure 2). The model is trained via minimizing a combined loss,

$$\ln\left(\frac{q(\boldsymbol{z}' \mid \boldsymbol{x}_I, \boldsymbol{x}_M)}{p(\boldsymbol{z}' \mid \boldsymbol{x}_I)}\right) - \ln p(\boldsymbol{x}_M \mid \boldsymbol{z}', \boldsymbol{x}_I) + \lambda * L_q\left(f(\boldsymbol{z}'), y\right) \tag{2}$$

where, $\boldsymbol{z}'$ is a random sample drawn from $q(\boldsymbol{z} \mid \boldsymbol{x}_I, \boldsymbol{x}_M)$ given the training datapoint $(\boldsymbol{x} = (\boldsymbol{x}_I, \boldsymbol{x}_M), y)$ and $\lambda$ is a hyper-parameter. The first term is the reconstruction loss and the second term regulates the shape of the posterior distribution, penalizing large deviations from the prior. The last term $L_q$ represents quantile loss in demand prediction. Once the model is trained, new samples can be generated as per the generative process.

Algorithm 1 outlines the pseudo-code for minimal edit selection. The algorithm starts with the current product and iteratively improves it by taking a small step in the latent space, in the direction of increasing business metric. Each gradient step ensures that the final product is not an outlier i.e. $p(\boldsymbol{x}'_M \mid \boldsymbol{x}_I)$ is sufficiently large and improves the demand. Note that $p(\boldsymbol{x}_M \mid \boldsymbol{x}_I)$ can be computed via Monte Carlo estimation i.e. $p(\boldsymbol{x}_M \mid \boldsymbol{x}_I) = \frac{1}{K}\sum_{i=1}^{K} p(\boldsymbol{x}_M \mid \boldsymbol{x}_I, z_i)$ where $z_{1\cdots K} \sim p(z \mid \boldsymbol{x}_I)$.

```
Algorithm 1: Algorithm for finding minimal edits
  Data: product $x = (x_I, x_M)$, encoder $q(z \mid x_I, x_M)$, decoder
        $p(x_M \mid z, x_I)$, max steps $N$, step size $\alpha$, decay $\beta < 1$
  Result: Minimal edits of $x$ (if any)
  $\alpha_0 \leftarrow \alpha$;
  $z_0 \sim q(z \mid x_I, x_M)$;
  $t \leftarrow 0$;
  MinimalEdits $\leftarrow \emptyset$;
  while $t \neq N$ do
      $z_t \leftarrow z_{t-1} + \alpha_t \cdot \left(\frac{\partial y}{\partial z}\right)_{z=z_{t-1}}$;
      sample $x'_M \sim p(x_M \mid z_t, x_I)$;
      if $(f(z_t) > f(z_0))$ and $\left(p(x'_M \mid x_I) \geq p(x_M \mid x_I)\right)$
       then
          MinimalEdits $\leftarrow$ MinimalEdits $\cup \{(x'_M, x_I)\}$;
      else
          $\alpha_{t+1} \leftarrow \alpha_t \cdot \beta$;
          $z_t \leftarrow z_{t-1}$;
      end
      $t \leftarrow t + 1$;
  end
  Return MinimalEdits
```

## 2.5 Scaling up Product Guidance

While it is theoretically possible to develop one model per product category, it is not scalable due to the sheer amount of maintenance required, for having a large number of models in a production system. Further, the model performance may suffer in categories with few products. On the other hand, categories that share attributes (e.g. *pajamas* and *trousers*), can be combined to improve the model performance. We use Spectral Co-Clustering algorithm to combine categories based on their attribute similarity and train a single model per cluster. The model is trained with category-id as an additional feature and a special token $< NA >$ is introduced in place of attributes that are irrelevant for a category.

## 3 EXPERIMENTS

We conduct a comprehensive set of experiments to gauge the effectiveness of our proposed algorithms. First, we compare the performance of the forecast model compared to several state-of-the-art baselines. We then conduct several ablations to quantify the impact of (a) clustering, (b) pre-training and (c) multi-task learning. Post that, we devise a framework to gauge the quality of product guidance insights generated by our system compared to other baselines. Finally, we conclude our experiments with qualitative and quantitative metrics on the effectiveness of minimal edit recommendations.

## 3.1 Dataset

We first describe the data-set used in our experiments. For each category, we extract the available products along with their attribute values, selling price and units sold in first one year after launch. The clustering is performed on three types of products: dresses, electronics and kitchenware. Following the clustering step, we train a model for each cluster. This reduces the number of models from several thousands to approximately 150. For the sake of brevity, we randomly select 5 clusters and present our analyses on them. For confidentiality reasons, we only refer to these category clusters with their IDs ($C_i$) without revealing details about their composition.

## 3.2 Experimental Setup

We train all the models on a randomly sampled data-set (70%) and tune the hyper-parameters on a validation data-set (10%). Finally, we report metrics on a held-out test set (20%). We compare PRODIGY to the following baselines: **Graph Attention Network (GAT)** [17], **FTTransformer (FTT)** [3], **SAINT** [16], **DeepFM** [4], **Deep and Cross Network (DCN)** [18] and **LightGBM** [7]. For GAT, we view our tabular data-set as a graph with each product, as well as attribute values represented as nodes and their relationship as edges. We chose LightGBM as it is quite popular for prediction tasks on tabular datasets. The target in our experiments is the number of units sold in the first 12 months after the launch of the product. Let the ground-truth values be $t_{1\ldots k}$ and predictions be $p_{1\ldots k}$. For each model, we report the following metrics for the regression task: a) ***RMSE***: The Root Mean Squared Error between true and predicted values b) ***WAPE***: Weighted Average Percentage Error is computed as $\frac{\sum_{i=1}^{k}|t_i - p_i|}{\sum_{i=1}^{k}|t_i|}$. While RMSE assigns equal importance to each sample, WAPE assigns higher importance to samples that have larger contribution to sales.

## 3.3 Results

*3.3.1 Baseline Comparison.* Table 1 summarizes our findings from this experiment. As the table shows, PRODIGY exhibits superior performance compared to other baselines across product clusters. We further observed that the performance improves with the increasing data-set size (refer to Figure 3). As the data-set size grows, PRODIGY outperforms all baselines by significant margin, 1-9% reduction in RMSE and 3-5% reduction in WAPE compared to FTT.

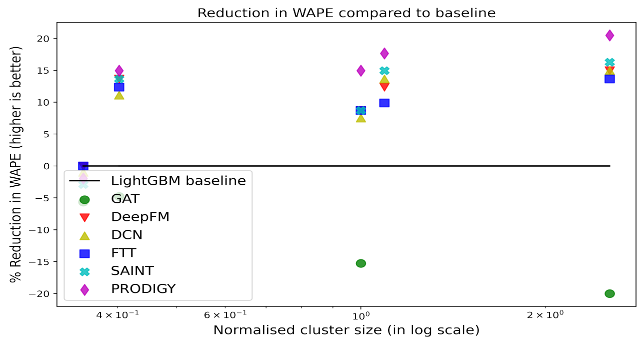

**Figure 3: Figure comparing performance of different models as the cluster size is varied (higher values are preferred).**

*3.3.2 Ablation Studies.* In this section, we present our findings from various ablation studies to quantify the impact of (a) clustering, (b) pre-training and (c) multi-task training.

**Table 1: Table showing RMSE and WAPE (the lesser the better) of various demand prediction models relative to LightGBM. The empty cells indicate experiment runs that failed due to out-of-memory issues.**

| Cluster | GAT | | DeepFM | | DCN | | FTT | | SAINT | | PRODIGY | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WAPE | RMSE | WAPE | RMSE | WAPE | RMSE | WAPE | RMSE | WAPE | RMSE | WAPE | RMSE |
| C0 | 1.055 | 1.001 | 0.884 | 0.846 | 0.897 | 0.857 | 0.888 | 0.867 | 0.884 | 0.858 | **0.866** | **0.840** |
| C1 | 1.254 | 1.096 | 0.871 | 0.788 | 0.870 | 0.797 | 0.876 | 0.819 | 0.863 | 0.799 | **0.832** | **0.757** |
| C2 | - | - | 0.893 | 0.858 | 0.879 | 0.856 | 0.908 | 0.902 | 0.874 | 0.856 | **0.855** | **0.819** |
| C3 | 1.185 | 1.115 | 0.921 | 0.903 | 0.929 | 0.924 | 0.920 | 0.936 | 0.915 | 0.923 | **0.868** | **0.862** |
| C4 | 1.058 | 1.009 | 1.004 | **0.963** | 1.006 | **0.955** | **1.000** | 0.969 | 1.027 | 0.973 | 1.015 | 0.968 |

**Effect of clustering:** We conduct an experiment where we train separate models for each category and compare its performance with the cluster-based model. In Figure 4, we show the results of this experiment. The results suggest that the cluster-based model achieves significant improvement over category-specific models (up to +15% reduction in RMSE, +12% reduction in WAPE). The improvement is pronounced for categories having fewer products and as the category size increases the gains are marginal.

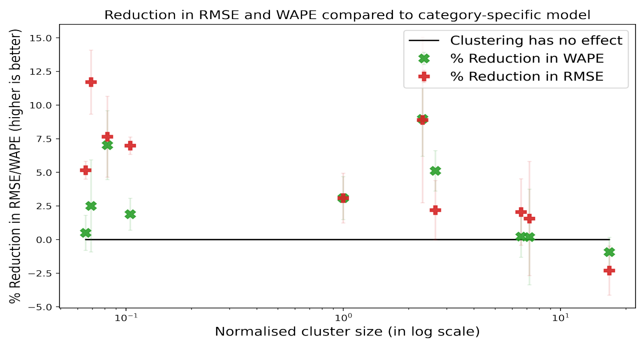

**Figure 4: Figure comparing performance of cluster-based models to category-specific models (higher values preferred).**

**Effect of pre-training:** For this set of experiments, we vary the masking rate and pre-train the model for 100 epochs. We try two variants of masking: using the same mask token across all attributes and using an attribute-specific mask token. In either case, we learn an embedding of mask token(s) during the pre-training stage. Table 2 shows the result of this experiment. We observe that using separate token for each attribute with mask rate of 0.2 results in 3% reduction in RMSE and 2.5% reduction in WAPE, highlighting the importance of pre-training. However, higher masking rate ($p > 0.2$) can wear off the benefits of pre-training, as the model learns spurious correlations in order to predict multiple missing values.

**Effect of multi-task training:** We also conduct an experiment where the demand prediction and the CVAE components of PRODIGY are trained separately, and compared to multi-task training using objective mentioned in Equation 2. Table 3 compares the loss on the test data-set for both these variations. We observe that multi-task training reduces the demand prediction loss by 2%, however, it increases the loss of the CVAE component by 14%. The overall loss reduces by 1% on multi-task training, due to lower contribution of latent and reconstruction losses.

**Table 2: Table showing percentage reduction in metrics due to pre-training compared to a model without any pre-training.**

| | Single token | | Separate tokens | |
|---|---|---|---|---|
| Probability | RMSE | WAPE | RMSE | WAPE |
| 0.15 | 0.9% | 0.6% | 2.8% | 1.5% |
| 0.2 | 1.2% | 0.8% | **3.0%** | **2.5%** |
| 0.25 | 0.7% | 0.3% | 2.7% | 1.2% |
| 0.3 | 0.2% | 0.2% | 1.8% | 0.6% |

**Table 3: Table comparing losses for multi-task training to when components are trained separately.**

| Loss Type | MAE Loss | CVAE Loss | Overall Loss |
|---|---|---|---|
| % Reduction | **2.1%** | -14.0% | **0.9%** |

### 3.3.3 *Recommending Minimal Edits.*

For recommending minimal edits, we run Algorithm 1 described in Section 2.4. We fix the category and the price band to condition the CVAE. We use a SGD optimizer for this task with the step-size set to 1e-5 and max steps set to 5000. Note that the output of the design search can be counter-factual, i.e. they do not currently exist on the e-commerce site. Hence, we do not have a golden data-set to evaluate the quality of recommendations. Therefore, we define the following metric and formulate a baseline to compare against. Given design $x$ as input, let $y$ be the recommended edit. We find $k$ products $x'_{1,2\cdots k}$ which are closest to $y$ in the test data-set, and compare their sales with $x$. We defined the fraction of $k$ nearest neighbors with improved sales figure as Precision@k, for the top-3 nearest neighbours of the Minimal Edit. We then define a baseline, called popularity-based search, where we find products with the highest predicted demand, at a maximum edit distance of 2 from the original design. We sample 30 products from each category, and find the minimal edit using PRODIGY and our baseline. Finally, we report the percentage of products where Precision@3 for PRODIGY was either higher than, lower than, or equal to the baseline. In Table 4, we highlight the result of this experiment.

**Table 4: Comparison of Precision@3 metric using PRODIGY and popularity-based search (baseline).**

| PRODIGY better | Both equal | Baseline better |
|---|---|---|
| 21.6% | 65.8% | 12.6% |

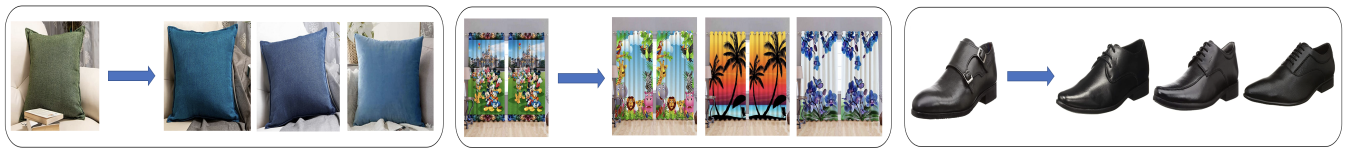

**Figure 5: Figure showing minimal edits for: a) cushion covers (left) b) curtains (center) and c) formal shoes (right).**

In Figure 6, we present an anecdote on the evolution of minimal edit for a product in sports shoe category, with an increasing number of steps. We observe that the minimal edit gradually changes from "red slip-on" to "bluish-grey lace-up".

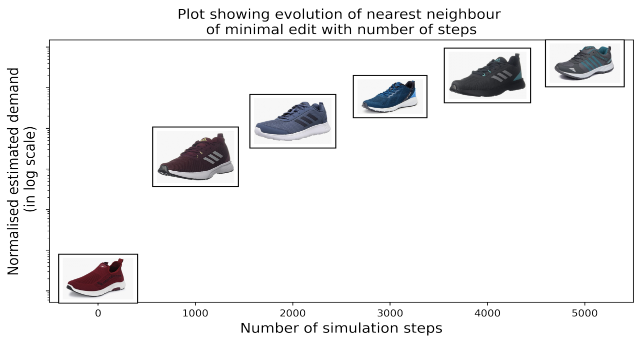

**Figure 6: Figure showing evolution of the minimal edit for a product in sports shoe category.**

*3.3.4 Evaluating Product Insights.* In this section, we compare the attribute rankings obtained based on Shapley values (c.f. Section 2.3) to a baseline ranking of attributes, defined by historical demand. We compare top attribute values recommended by PRODIGY against the values recommended by baseline and observe their future demand, i.e. average demand in the next 3 months following the observation period. In Table 5, we highlight the result of this experiment. We observe that for 74% of the attributes, baseline and PRODIGY recommended the same set of top attribute values. However, PRODIGY performs better for 16% attributes compared to 10% for the baseline policy, which indicates that top attribute values recommended by PRODIGY are more indicative of higher sales.

**Table 5: Comparison of attribute ranking from PRODIGY model vs baseline ranking obtained using historical demand.**

| Cluster | PRODIGY better | Baseline better | Both equal |
|---|---|---|---|
| **C0** | 24% | 16% | 60% |
| **C1** | 20% | 14% | 54% |
| **C2** | 18% | 10% | 56% |
| **C3** | 16% | 6% | 62% |
| **C4** | 12% | 10% | 72% |
| **Overall** | 16% | 10% | 74% |

*3.3.5 Online Experiments.* To evaluate the quality of recommendations and design intelligence, we built a dashboard powered by PRODIGY across all categories and released it to a random cohort of sellers. The treatment group consists of sellers who had access to the dashboard and the control group is the remaining cohort of sellers. We track new products launched by the treatment group and compare them to products (from the same category) launched by the control group. The products are evaluated on product ratings, page views and sales throughput. We observe that across categories, PRODIGY recommendations drive 40% higher page views, 20% higher sales and 1.3% improvement in product rating.

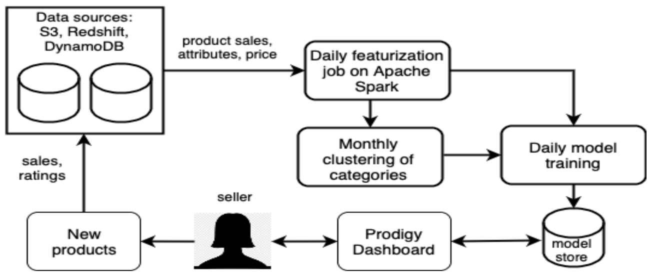
Data sources: S3, Redshift, DynamoDB
product sales, attributes, price
Daily featurization job on Apache Spark
Monthly clustering of categories
Daily model training
sales, ratings
seller
New products
Prodigy Dashboard
model store

**Figure 7: High-level architecture of PRODIGY system.**

## 4 RELATED WORKS

To the best of our knowledge, there is no published work on generating product design insights and guidance at scale. Previous research [6, 19] have focused on extracting qualitative insights such as *battery life, easy to clean,* etc. and sentiment from customer reviews. However, qualitative insights are not directly actionable, as brands can only control product attributes. Another related line of work on generating aesthetic designs for fashion [5, 13] and automobile [12]. However, their work is limited to aesthetic features such as texture and pattern. Another related area is activation maximization [11], which is a gradient-based method that optimizes the input to a neural network to highly activate a target neuron. It is widely used to explain the behavior of a neural network [10, 14] and synthesize images [5] with improved fashion-ability. However, naively using activation maximization can lead to infeasible products, which we counter through a data-density approximator.

## 5 CONCLUSION AND FUTURE WORK

In this paper we present PRODIGY, a system for mining product guidance insights for brands of a major e-commerce platform. We develop novel algorithms to estimate demand forecasts for product designs and recommend minimal attribute changes to improve performance of existing products. Through a convincing set of experiments, we demonstrate the efficacy and efficiency of our techniques. In the future, we plan to extend the forecast model to provide a confidence interval, and incorporate more constraints on the minimal edits e.g. *"find attribute changes that improve sales by +X% and rating by +Y%".*

## REFERENCES

[1] 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* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[3] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting Deep Learning Models for Tabular Data. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 18932–18943. https://proceedings.neurips.cc/paper/2021/file/9d86d83f925f2149e9edb0ac3b49229c-Paper.pdf

[4] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. *CoRR* abs/1703.04247 (2017). arXiv:1703.04247 http://arxiv.org/abs/1703.04247

[5] Wei-Lin Hsiao, Isay Katsman, Chao-Yuan Wu, Devi Parikh, and Kristen Grauman. 2019. Fashion++: Minimal Edits for Outfit Improvement. *CoRR* abs/1904.09261 (2019). arXiv:1904.09261 http://arxiv.org/abs/1904.09261

[6] Jiaxin Huang, Yu Meng, Fang Guo, Heng Ji, and Jiawei Han. 2020. Weakly-Supervised Aspect-Based Sentiment Analysis via Joint Aspect-Sentiment Topic Embedding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6989–6999. https://doi.org/10.18653/v1/2020.emnlp-main.568

[7] 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 *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. 3149–3157.

[8] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-attentive Sentence Embedding. *CoRR* abs/1703.03130 (2017). arXiv:1703.03130 http://arxiv.org/abs/1703.03130

[9] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 4765–4774. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[10] Aravindh Mahendran and Andrea Vedaldi. 2016. Visualizing Deep Convolutional Neural Networks Using Natural Pre-Images. *Int. J. Comput. Vision* 120, 3 (dec 2016), 233–255. https://doi.org/10.1007/s11263-016-0911-8

[11] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. 2016. Synthesizing the Preferred Inputs for Neurons in Neural Networks via Deep Generator Networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (Barcelona, Spain) *(NIPS'16)*. Curran Associates Inc., Red Hook, NY, USA, 3395–3403.

[12] Yanxin Pan, Alexander Burnap, Jeffrey Hartley, Richard Gonzalez, and Panos Y. Papalambros. 2017. Deep Design: Product Aesthetics for Heterogeneous Markets. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3097983.3098176

[13] Abhinav Ravi, Arun Patro, Vikram Garg, Anoop Kolar Rajagopal, Aruna Rajan, and Rajdeep Hazra Banerjee. 2019. Teaching DNNs to design fast fashion. https://doi.org/10.48550/ARXIV.1906.12159

[14] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *CoRR* abs/1312.6034 (2013). http://dblp.uni-trier.de/db/journals/corr/corr1312.html#SimonyanVZ13

[15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning Structured Output Representation using Deep Conditional Generative Models. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf

[16] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C. Bayan Bruss, and Tom Goldstein. 2021. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training. *CoRR* abs/2106.01342 (2021). arXiv:2106.01342 https://arxiv.org/abs/2106.01342

[17] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. *6th International Conference on Learning Representations* (2017).

[18] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-Scale Learning to Rank Systems *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 1785–1797. https://doi.org/10.1145/3442381.3450078

[19] Watanabe Yusuke, Hrushka Estevam, Ginuga Karthik, Porter Danielle, and Agarwal Saurabh. 2019. Product Attribute Discovery from Review Texts. In *AMLC*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf