

# ML-based Visualization Recommendation: Learning to Recommend Visualizations from Data

Xin Qian

University of Maryland  
College Park, MD, USA

Eunyee Koh

Adobe Research  
San Jose, CA, USA

Ryan A. Rossi

Adobe Research  
San Jose, CA, USA

Sana Malik

Adobe Research  
San Jose, CA, USA

Fan Du

Adobe Research  
San Jose, CA, USA

Sungchul Kim

Adobe Research  
San Jose, CA, USA

Joel Chan

University of Maryland  
College Park, MD, USA

## ABSTRACT

Visualization recommendation seeks to generate, score, and recommend to users useful visualizations automatically, and are fundamentally important for exploring and gaining insights into a new or existing dataset quickly. In this work, we propose the first end-to-end ML-based visualization recommendation system that takes as input a large corpus of datasets and visualizations, learns a model based on this data. Then, given a new unseen dataset from an arbitrary user, the model automatically generates visualizations for that new dataset, derive scores for the visualizations, and output a list of recommended visualizations to the user ordered by effectiveness. We also describe an evaluation framework to quantitatively evaluate visualization recommendation models learned from a large corpus of visualizations and datasets. Through quantitative experiments, a user study, and qualitative analysis, we show that our end-to-end ML-based system recommends more effective and useful visualizations compared to existing state-of-the-art rule-based systems. Finally, we observed a strong preference by the human experts in our user study towards the visualizations recommended by our ML-based system as opposed to the rule-based system (5.92 from a 7-point Likert scale compared to only 3.45).

## KEYWORDS

Visualization recommendation, learning-based visualization recommendation, data visualization, machine learning, deep learning

## 1 INTRODUCTION

Nowadays, visualization has been a convenient vehicle for exploratory data analysis. However, due to the increasing size of real-world datasets, there are sometimes obstacles for practitioners, such as decision makers, data analysts, and researchers, to efficiently and effectively create visualizations. It could be overwhelming to understand an unfamiliar dataset, then select the most proper visualizations out of a myriad of valid visualization choices. Automatic visualization recommendation systems have been developed to assist data analysts in creating visualizations. An end-to-end visualization recommendation system would automatically recommend a list of visualizations ordered by importance, where the visualizations uses the proper visual design to show insights about a selection of attributes<sup>1</sup> in the dataset. A successful system would

<sup>1</sup>The term variable, attribute, and data column are synonyms in this work.

greatly reduce the amount of time, cost, and effort that human spend in insight discovery process.

Previous end-to-end systems are rule-based, and leverage a small set of manually defined rules crafted by domain experts to score the generated visualizations [16, 18]. As such, these rule-based systems have many issues that our proposed approach addresses. First, these systems often have quality issues, limiting the utility and usefulness of the recommended visualizations. Second, when visualizations are scored using a set of manually defined rules/heuristics, many of the visualizations receive the same exact score. This issue arises due to the way scoring is done using the rules, which often simply assigns a positive (or negative) score depending on the rule they abide by or violate, and at the end, all such scores from the small set of rules that actually apply to the visualization are combined to obtain the final score, which results in many visualizations having the same score, and thus, unable to differentiate between the visualizations receiving the same heuristic score. Third, adding additional rules to these systems is tedious and costly in terms of time and effort required. Finally, in contrast to our proposed approach that is completely automatic and data-driven, and able to adapt based on new data, or user-preferences, the existing rule-based systems are not automatic, not data-driven, and hard to iterate or improve as the preferences of users and visualizations change over time.

While there are only a few such end-to-end visualization recommendation systems, all of them are fundamentally rule-based [16, 18]. Previous work such as VizML [3] have used machine learning to predict the type of a chart (e.g., bar, scatter) instead of complete visualization, whereas other work such as Draco [9] used a model to infer weights for a set of manually defined rules. There is no such work that uses machine learning for end-to-end visualization recommendation<sup>2</sup> in a completely automated and data-driven fashion. This paper fills this gap by proposing the first completely automated and data-driven approach for end-to-end ML-based visualization recommendation.

In this work, we propose *the first end-to-end deep learning-based visualization recommendation system* that automatically learns to generate effective and useful visualizations by leveraging previous user-generated visualizations as training. Suppose a user selects or uploads a new dataset of interest, we can then use the learned model

<sup>2</sup>Note that other work used visualization recommendation more generally, however, in this work the term visualization recommendation has a very precise and formal definition to mean the recommendation of an actual visualization, not only simple design choices like chart type [3], or weights for manually defined rules [9], etc.

$\mathcal{M}$  to automatically score and recommend to the user the top-k most relevant visualizations for the users' dataset. The approach is completely automatic, fully data-driven, flexible, and effective. It is able to learn a general recommendation model  $\mathcal{M}$  from a large corpus of datasets and their visualizations, which can then be applied for scoring and recommending visualizations for any other arbitrary dataset.

We first formalize the ML-based visualization recommendation problem and describe a general learning framework for it. To learn a visualization recommendation model from a large corpus of training visualizations, we decompose a visualization into the subset of attributes selected from one of the datasets in the training corpus and a visualization configuration that describes the design choices and types of attributes required. In particular, the proposed notion of a visualization configuration represents a data-independent abstraction where the data attributes used in the design choices are replaced by their general type. Both of these provide us with everything required to characterize a visualization. Next, we propose a wide-and-deep learning model for visualization recommendation based on this problem formulation that learns from the attribute selections and their visualization configurations. In the wide component, we learn from sparse attribute meta-features along with sparse visualization configuration features whereas in the deep component we learn from dense representations of the meta-features of the attributes and visualization configurations. Scores from both these components and then combined to obtain the final score of a complete visualization.

Many new evaluation issues and challenges arise when quantitatively evaluating the ranking of visualizations from an end-to-end trained ML-based visualization recommendation model. For instance, since visualizations held-out for evaluation are from different datasets, and the space of possible visualizations to recommend differs for each dataset (as shown in Sec. 3), then standard ranking evaluation metrics fail since the size of the visualization space changes depending on the data. Consider two datasets, one with a large number of attributes and another with only two such attributes, then the *ML-based visualization ranking problem* in the dataset with a few attributes is significantly easier than the one with a large number of attributes. To address these challenges, we propose a general framework for evaluation of end-to-end ML-based visualization recommendation system. This is the first evaluation framework for ML-based visualization recommender systems, and as such, we believe it will be useful for making further progress in developing better and more accurate end-to-end visualization recommendation systems that leverage machine learning. The evaluation framework serves as a foundation for quantitative evaluation of future ML-based vis. rec. systems that build upon our work.

Extensive experiments evaluating the effectiveness of our approach are provided in the paper. Overall, the results demonstrate the effectiveness and utility of the proposed end-to-end ML-based visualization recommendation system. First, we conduct extensive experiments using a large-scale public corpus of datasets and user-generated visualizations. Our empirical results demonstrate the effectiveness of our approach as it is able to recover the held-out ground-truth visualizations among the large exponential space of lower quality/irrelevant visualizations. We also conduct a user

study to investigate the quality of our wide-and-deep learning-based end-to-end visualization recommendation system compared to the state-of-the-art rule-based system. In nearly all cases, the visualizations generated and recommended by our end-to-end ML-based system are at least as good, and most often, better than those recommended by the rule-based system. Furthermore, we demonstrate the effectiveness of our approach through a number of case studies that clearly show a variety of important advantages of our approach compared to the existing rule-based system.

## Main Contributions

A summary of the main contributions of this work are as follows:

- **First ML-based Visualization Recommender:** In this work, we propose the first end-to-end *ML-based visualization recommendation system* that automatically learns a model from a large set of  $N$  training datasets and the corresponding  $N$  sets of user-generated visualizations. The learned model not only captures simple visual rules, but is able to learn complex high-dimensional latent characteristics behind effective user-generated visualizations from the training corpus along with the latent characteristics of the data (subset of attributes) that are associated with the visualizations. Given a new (unseen) dataset of interest, our learned model can then generate, score, and automatically recommend the top most insightful and effective visualizations for this new dataset.<sup>3</sup>
- **Problem Formulation:** We carefully formalize the problem of learning a visualization recommendation model from training data consisting of  $N$  datasets and  $N$  sets of visualizations.<sup>4</sup> To the best of our knowledge, this is the first formal presentation of the ML-based vis. rec. problem.
- **Evaluation Framework:** We describe a general framework for evaluation of end-to-end ML-based visualization recommendation systems using a held-out set of known ground-truth visualizations from a set of new held-out datasets that were not used to train the model. The evaluation framework serves as a foundation for quantitative evaluation of future ML-based vis. rec. systems that build upon our work.
- **Effectiveness:** We demonstrate the effectiveness of our approach through a comprehensive set of experiments including quantitative evaluation of the visualization ranking (Sec. 6.1), user study comparing our ML-based system to a rule-based system (Sec. 6.2), and a qualitative case study (Sec. 6.3). Overall, the results demonstrate the effectiveness and utility of the proposed end-to-end ML-based visualization recommendation system.

## 2 RELATED WORK

Related work can be categorized as follows: (i) rule-based systems that recommend entire visualizations, and (ii) approaches designed for simpler, but fundamentally different sub-tasks such as predicting the chart type of a visualization. Despite that they do not solve the end-to-end visualization recommendation problem, we include them since some of them use machine learning to solve a fundamentally different problem.

<sup>3</sup>Note each visualization uses a subset of the attributes from the dataset, and some attributes may never be used.

<sup>4</sup>A single dataset is associated with a set of visualizations from that dataset.

## ML-based Visualization Recommendation: Learning to Recommend Visualizations from Data

Systems that recommend entire visualizations to users have existed since 1980s [5]. Early systems include Automatic Presentation Tool (APT) [7] that generates graphical presentations from data, SAGE [11] that uses a search algorithm to select and compose graphics from data, and ShowMe [8]. These systems can be attributed as rule-based solutions. For example, Automatic Presentation Tool (APT) [7] uses logical statements of visualization design knowledge that come from human perceptual experiments. SAGE [11] generates visualizations based on partial specifications. ShowMe [8] offers users with a set of defaults to filter valid visualization types based on characteristics of selected data. Mixed-initiative systems such as Voyager [13, 16, 18], VizDeck [10], and DIVE [4] incorporate visual encoding rules to assist user data exploration data exploration. Rule-based systems have many limitations that our work addresses. For instance, rule-based recommendation systems rely entirely on a large set of manually defined rules from domain experts, which are costly in terms of the manual labor required, and may miss many important rules that would provide users with significantly more effective visualizations for their dataset of interest. Such approaches are clearly not data-driven and difficult to adapt as one would need to routinely incorporate new rules in a manual fashion, which is costly in terms of the time and effort required by domain experts to maintain such systems. Further, rules for such systems need to be manually defined with respect to the domain of interest. For instance, visualizations for data scientists, or scientific domains are likely different from visualizations that journalists prefer or those that would work well for elderly populations. Therefore, new rule sets would likely be required to effectively recommend visualizations for each group. These systems also require tailored experiments with human users to validate the manually defined rules. In comparison, our work learns a model  $\mathcal{M}$  to recommend entire visualizations directly from a large corpus of training data, in a fully automatic data-driven fashion. Furthermore, we also propose an evaluation framework to validate the effectiveness of ML-based visualization recommendation models.

On the other hand, there are systems that tackle sub-tasks in visualization recommendation. Each of those systems has a distinct focus in some end goals such as improving expressiveness, improving perceptual effectiveness, matching user task types, etc. The sub-tasks can generally be divided two categories [5, 17]: whether the solution focuses on recommending data (*what data to visualize*), such as Discovery-driven Data Cubes [12], Scagnostics [14], AutoVis [15], and MuVE [1]) or recommending encoding (*how to design and visually encode the data*), such as APT [7], ShowMe [8], and Draco-learn [9]). While some of those are ML-based, none recommends entire visualizations, and thus does not solve the visualization recommendation problem that lies at the heart of our work. For example, VizML [3] used machine learning to predict the type of a chart (e.g., bar, scatter, etc.) instead of complete visualization. Another work Draco [9] used a model to infer weights for a set of manually defined rules. VisPilot [6] recommended different drill-down data subsets from datasets. Instead of solving simple sub tasks such as predicting the chart type of a visualization, we focus on the *end-to-end visualization recommendation task* where the goal is to *automatically recommend users the top-k most effective visualizations as the output, given an input dataset from the user*.

This paper fills the gap by proposing the first end-to-end ML-based visualization recommendation approach that is completely automatic and data-driven. It tackles both choosing data from datasets, and recommending encoding for selected data, therefore achieving the goal of recommending complete visualizations from arbitrary datasets using an automatically learned model  $\mathcal{M}$  from a large corpus of training data.

### 3 ML-BASED PROBLEM FORMULATION

In this section, we formally introduce the ML-based visualization recommendation problem, and present a generic learning framework for it, which includes two key parts:

- **Model Training (Sec. 3.1):** Given a training visualization corpus  $\mathcal{D} = \{\mathbf{X}_i, \mathbb{V}_i\}_{i=1}^N$  consisting of  $N$  datasets  $\{\mathbf{X}_i\}_{i=1}^N$  and the corresponding  $N$  sets of visualizations  $\{\mathbb{V}_i\}_{i=1}^N$ <sup>5</sup>, we first learn a model  $\mathcal{M}$  from the training corpus  $\mathcal{D}$  that best captures and scores effective visualizations highly and assigns low scores to bad/ineffective visualizations.<sup>6</sup>
- **Recommending Visualizations (Sec. 3.2):** Given a new (unseen) dataset  $\mathbf{X}_{\text{test}} \notin \mathcal{D}$  of interest, our learned visualization recommendation model  $\mathcal{M}$  is used to generate, score, and automatically recommend the top most insightful and effective visualizations for this new dataset.<sup>7</sup>

Notice that the fundamental difference between the rule-based visualization recommendation problem and our proposed ML-based visualization recommendation problem is that ML-based models are automatically learned from data whereas rule-based approaches are manually defined (and are not true models).

The visualization recommendation training data  $\mathcal{D} = \{\mathbf{X}_i, \mathbb{V}_i\}_{i=1}^N$  can be general, as it can consist of a set of datasets and a set of relevant visualizations from each dataset collected from a variety of different sources.<sup>8</sup> For instance, the corpus may consist of datasets and visualizations collected from websites (e.g., by crawling the web) or from a visual analytic platform such as Tableau and Power BI where users upload datasets and created corresponding visualizations. Depending on the corpus, the definition of a visualization to be effective is also flexible and that reflects how users from that corpus source perceive as effective visualizations. For example, a visualization in a data journalism website emphasizes attractiveness while a visualization in scientific papers need to be straightforward and scientifically meaningful. We use the corpus to train the ML-based vis. rec. model.

Each visualization uses a subset of attributes from a dataset  $\mathbf{X}_i$ , which we call the subset as the *attribute combination*. We now define the space of attribute combinations  $\mathcal{X}_i = \{\mathbf{X}_i^1, \dots, \mathbf{X}_i^{(k)}, \dots\}$  for an arbitrary dataset  $\mathbf{X}_i$ , which can be either a training dataset  $\mathbf{X}_i \in \mathcal{D}$  or a new test dataset  $\mathbf{X}_{\text{test}} \leftarrow \mathbf{X}_i \notin \mathcal{D}$ .

<sup>5</sup> $\mathbb{V}_i$  is the set of visualizations associated with the  $i$ th dataset  $\mathbf{X}_i$ .

<sup>6</sup>The learned model  $\mathcal{M}$  not only captures simple visual rules, but is able to learn complex high-dimensional latent characteristics behind effective user-generated visualizations from the training corpus along with the latent characteristics of the data (subset of attributes) that are associated with the visualizations.

<sup>7</sup>Note each visualization uses a subset of the attributes from the dataset, and some attributes may never be used.

<sup>8</sup>Hence, each dataset has a corresponding set of visualizations that use a subset of attributes from the dataset.

**DEFINITION 1 (SPACE OF ATTRIBUTE COMBINATIONS).** Given an arbitrary dataset matrix  $X_i$ , let  $\mathcal{X}_i$  denote the space of attribute combinations of  $X_i$  defined as

$$\Sigma : X_i \rightarrow \mathcal{X}_i, \quad s.t. \quad (1)$$

$$\mathcal{X}_i = \{X_i^{(1)}, \dots, X_i^{(k)}, \dots\}, \quad (2)$$

where  $\Sigma$  is an attribute combination generation function and every  $X_i^{(k)} \in \mathcal{X}_i$  is a different subset (combination) of attributes from  $X_i$ , and thus  $X_i^{(k)}$  may consist of one, two, or more attributes from  $X_i$ .

When the dataset  $X_i$  is a new test dataset  $X_i \notin \mathcal{D}$ , we use  $X_{\text{test}}$  to denote the new unseen dataset and the space of attribute combinations from the new test dataset is  $\mathcal{X}_{\text{test}}$ .

Let  $|X_i|$  and  $|X_j|$  denote the number of attributes (columns) of two arbitrary datasets  $|X_i|$  and  $|X_j|$ , respectively.

**PROPERTY 1.** If  $|X_i| > |X_j|$ , then  $|\mathcal{X}_i| > |\mathcal{X}_j|$ .

The proof of Property 1 is straightforward, but Property 1 will be important later when characterizing the space of possible visualizations from a given dataset.

**DEFINITION 2 (SPACE OF VISUALIZATION CONFIGURATIONS).** Let  $\mathcal{C}$  denote the space of all visualization configurations such that a visualization configuration  $C_{ik} \in \mathcal{C}$  defines an abstraction of a visualization where for each visual design choice ( $x$ ,  $y$ , marker-type, color, size, etc.) that maps to an attribute in  $X_i$ , we replace it with its type. Therefore visualization configurations are essentially visualizations without any attribute (data).

**PROPERTY 2.** Every visualization configuration  $C_{ik} \in \mathcal{C}$  is independent of any data matrix  $X$  (by Definition 2).

The above implies that  $C_{ik} \in \mathcal{C}$  can potentially arise from any arbitrary dataset and is therefore not tied to any specific dataset since visualization configurations are general abstractions where the data bindings have been replaced with their general type (e.g., if  $x/y$  in some visualization mapped to an attribute in  $X_i$ , then it is replaced by the type of that attribute, that is, ordinal, quantitative, categorical, etc.).

A visualization configuration and the attributes selected<sup>9</sup> is everything necessary to generate a visualization. See Figure 1 for an example. The size of the space of visualization configurations is large since visualization configurations come from all possible combinations of design choices and their values such as,

- **mark/chart:** bar, scatter, ...
- **x-type:** quantitative, nominal, ordinal, temporal, ..., none
- **y-type:** quantitative, nominal, ordinal, temporal, ..., none
- **color:** red, ..., quantitative, nominal, ordinal, temporal, ...
- **size:** 1pt, 2pt, ..., quantitative, nominal, ordinal, temporal, ...
- **x-aggregate:** sum, mean, bin, ..., none
- **y-aggregate:** sum, mean, bin, ..., none
- ...

Recall that  $\Sigma$  is an attribute combination generation function defined as  $\Sigma : X_i \rightarrow \mathcal{X}_i$  where  $\mathcal{X}_i$  is the space of all combinations of attributes from dataset  $X_i$ , i.e., all subsets of one or more

<sup>9</sup>Selected attributes is the same as the combination of attributes defined in Eq. 2.

attributes from  $X_i$ . For instance, suppose we have a dataset with three attributes  $X = [x_1 \ x_2 \ x_3]$ , then  $\Sigma(X) = \mathcal{X}$  is:

$$\Sigma(X) = \left\{ \underbrace{x_1, x_2, x_3}_{\Sigma_1(X)}, \underbrace{[x_1 \ x_2], [x_1 \ x_3], [x_2 \ x_3]}_{\Sigma_2(X)}, \underbrace{[x_1 \ x_2 \ x_3]}_{\Sigma_3(X)} \right\} \quad (3)$$

**DEFINITION 3 (SPACE OF VISUALIZATIONS OF  $X_i$ ).** Given an arbitrary dataset matrix  $X_i$ , we define  $\mathbb{V}_i^*$  as the space of all possible visualizations that can be generated from  $X_i$ . More formally, the space of visualizations  $\mathbb{V}_i^*$  is defined with respect to a dataset  $X_i$  and the space of visualization configurations  $\mathcal{C}$ ,

$$\Sigma(X_i) = \mathcal{X}_i = \{X_i^1, \dots, X_i^{(k)}, \dots\} \quad (4)$$

$$\xi : \mathcal{X}_i \times \mathcal{C} \rightarrow \mathbb{V}_i^* \quad (5)$$

where  $\mathcal{X}_i = \{X_i^1, \dots, X_i^{(k)}, \dots\}$  is the set of all possible attribute/attribute combinations of  $X_i$  (Def. 1). More succinctly,  $\xi : \Sigma(X_i) \times \mathcal{C} \rightarrow \mathbb{V}_i^*$ , and therefore  $\xi(\Sigma(X_i), \mathcal{C}) = \mathbb{V}_i^*$ .

In other words, given an attribute combination  $X_i^{(k)} \in \mathcal{X}_i$  consisting of a subset of attributes from dataset  $X_i$  and a visualization configuration  $C \in \mathcal{C}$ , then  $\xi(X_i^{(k)}, C)$  is the corresponding visualization. Define  $X \neq Y \implies \forall i, j \ x_i \neq y_j$ .

**LEMMA 1.**  $\forall X_i, X_j$  s.t.  $X_i \neq X_j$ , then  $\xi(\Sigma(X_i), \mathcal{C}) \cap \xi(\Sigma(X_j), \mathcal{C}) = \emptyset$ .

This is straightforward to see and implies that when  $\mathcal{C}$  is fixed, the space of visualizations is entirely dependent on the dataset, and for any two datasets  $X_i$  and  $X_j$  without any shared attributes/overlap  $X_i \neq X_j$ , then the set of possible visualizations that can be generated from either dataset are entirely disjoint from one another, that is  $\mathbb{V}_i = \xi(\Sigma(X_i), \mathcal{C})$  and  $\mathbb{V}_j = \xi(\Sigma(X_j), \mathcal{C})$  where  $\mathbb{V}_i \cap \mathbb{V}_j = \emptyset$ . Hence,  $|\mathbb{V}_i \cup \mathbb{V}_j| = 0$  and  $|\mathbb{V}_i \cup \mathbb{V}_j| = |\mathbb{V}_i| + |\mathbb{V}_j|$ . If  $|X_i| > |X_j|$ , then  $|\xi(\Sigma(X_i), \mathcal{C})| > |\xi(\Sigma(X_j), \mathcal{C})|$ .

**DEFINITION 4 (POSITIVE VISUALIZATIONS OF  $X_i$ ).** Given an arbitrary dataset matrix  $X_i$ , we define  $\mathbb{V}_i$  as the set of positive visualizations (user-generated, observed) from dataset  $X_i$ . Therefore,

$$\mathbb{V} = \bigcup_{i=1}^N \mathbb{V}_i \quad \text{and} \quad |\mathbb{V}| \geq N \quad (6)$$

**DEFINITION 5 (NEGATIVE VISUALIZATIONS OF  $X_i$ ).** Let  $\mathbb{V}_i^*$  denote the space of all visualizations that arise from the  $i$ th dataset  $X_i$  such that the user-generated (positive) visualizations  $\mathbb{V}_i$  satisfies  $\mathbb{V}_i \subseteq \mathbb{V}_i^*$ , then the space of negative visualizations for dataset  $X_i$  is

$$\mathbb{V}_i^- = \mathbb{V}_i^* \setminus \mathbb{V}_i \quad (7)$$

This follows from  $\mathbb{V}_i^- \cup \mathbb{V}_i = \mathbb{V}_i^*$ .

**NOTE.** The space of negative visualizations between different datasets is also obviously completely disjoint.

Given  $\mathbb{V}_i^*$  as the space of all visualizations of the  $i$ th dataset  $X_i$  where it consists of positive and negative visualizations, denoted as  $\mathbb{V}_i^- \cup \mathbb{V}_i = \mathbb{V}_i^*$ , we define  $Y_{ik}$  as the ground-truth label of a visualization  $V_{ik} \in \mathbb{V}_i^*$ , such that

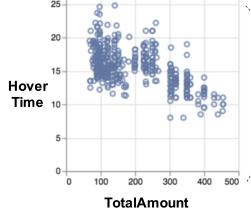
$$Y_{ik} = \begin{cases} 1 & \text{if } V_{ik} \in \mathbb{V}_i \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

**ML-based Visualization Recommendation:**  
Learning to Recommend Visualizations from Data

A dataset with **many** attributes (top) and **one of** its visualizations (bottom left)

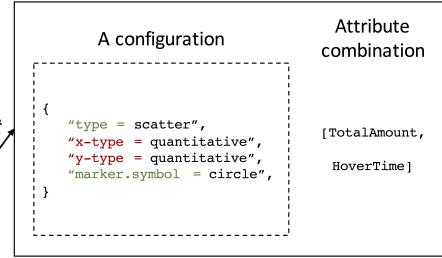
ID	Timestamp	HoverTime	Currency	Browser	Disp	...	TotalAmount
2E073CA78 5310000	2018/12/10 14:33	18	USD	Mozilla (iPhone)	en-gb		121
2E073CA78 5310001	2018/12/10 14:15	15	EUR	Chrome (Mac OS)	de-DE		97
...							
2E073CA78 5320000	2018/12/10 14:36	31	GBP	Safari (iPhone)	en-US		238

The visualization uses a **subset of** attributes, HoverTime and TotalAmount



```
"type = scatter",
"x = TotalAmount",
"y = HoverTime",
"marker.symbol = circle"
```

A positive visualization



**Figure 1: The process of extracting positive training visualizations.** The left figure shows a dataset from the corpus. The dataset has a set of visualizations. One visualization uses a subset of attributes from the dataset. The right figure is an extracted positive visualization that separates the visualization into a configuration and attribute selection. The visualization will be used for training the visualization recommendation model.

**DEFINITION 6 (SAMPLING NEGATIVE VISUALIZATIONS OF  $X_i$ ).**  
Given dataset  $X_i$ , we sample negative visualizations from  $\mathbb{V}_i^- = \mathbb{V}_i^* \setminus \mathbb{V}_i$  (Def. 5) as follows:

$$k \sim \text{UniformDiscrete}\{1, 2, \dots, |\mathbb{V}_i^-|\}, \quad \text{for } j = 1, 2, \dots \quad (9)$$

$$\widehat{\mathbb{V}}_i^- = \widehat{\mathbb{V}}_i \cup \mathcal{V}_{ik}^- \quad (10)$$

where  $\widehat{\mathbb{V}}_i^- \subseteq \mathbb{V}_i^-$ . Hence,  $\mathcal{V}_{ij}^-$  denotes the  $j$ th negative visualization for dataset  $X_i$  sampled from  $\mathbb{V}_i^- \in \mathbb{V}_i$ .

The negative visualization space is large and therefore sampling of this vast space is required to ensure fast and computationally tractable inference. In Eq. 9, we sample the negative visualization space of dataset  $X_i$  uniformly at random with replacement. Recall that any form of estimation is difficult since the size of the space of visualizations  $\mathbb{V}_i^*$ , including positive visualizations  $\mathbb{V}_i$  and negative visualizations  $\mathbb{V}_i^-$  depends entirely on the number of attributes/attributes in the dataset  $X_i$  (and their types, such as real-valued, ordinal, categorical, etc.) used in their generation, and thus the size of the different visualization spaces varies based on it. Sampling negative visualizations is important for both training and testing.

### 3.1 Learning Vis. Rec. Model

Now we formulate the problem of training a visualization recommendation model  $\mathcal{M}$  from a large training corpus of datasets and sets of visualizations associated to each dataset.

**DEFINITION 7 (LEARNING VIS. RECOMMENDATION MODEL).**  
Let  $\mathcal{D} = \{X_i, \mathbb{V}_i\}_{i=1}^N$  denote the training set consisting of datasets  $\{X_i\}_{i=1}^N$  and the corresponding  $N$  sets of visualizations  $\{\mathbb{V}_i\}_{i=1}^N$  for the  $N$  datasets. Given the set of training datasets and relevant visualizations  $\mathcal{D} = \{X_i, \mathbb{V}_i\}_{i=1}^N$ , the goal is to learn a visualization

recommendation model  $\mathcal{M}$  by optimizing the following general objective function,

$$\underset{\mathcal{M}}{\operatorname{argmin}} \sum_{i=1}^N \sum_{(X_i^{(k)}, C_{ik}) \in \mathbb{V}_i^- \cup \mathbb{V}_i} \mathcal{L}(Y_{ik} | \Psi(X_i^{(k)}), f(C_{ik}), \mathcal{M}) \quad (11)$$

where  $\mathcal{L}$  is the loss function,  $Y_{ik} = \{0, 1\}$  is the ground-truth label of the  $k$ th visualization  $\mathcal{V}_{ik} = (X_i^{(k)}, C_{ik}) \in \mathbb{V}_i^- \cup \mathbb{V}_i$  for dataset  $X_i$ . Further,  $X_i^{(k)} \subseteq X_i$  is the combination of attributes used in the visualization. In Eq. 11,  $\Psi$  and  $f$  are general functions over the attribute combination  $X_i^{(k)} \subseteq X_i$  and the visualization configuration  $C_{ik}$  of the visualization  $\mathcal{V}_{ik} = (X_i^{(k)}, C_{ik}) \in \mathbb{V}_i^- \cup \mathbb{V}_i$ , respectively.<sup>10</sup>

For computational tractability, we replace  $\mathbb{V}_i^-$  in Eq. 11 with the set  $\widehat{\mathbb{V}}_i^-$  of sampled negative visualizations for the  $i$ th dataset matrix  $X_i$ . As an aside, we provide a general formulation of the training of the model  $\mathcal{M}$  in Definition 7. Intuitively, the learned model  $\mathcal{M}$  from Eq. 11 can then be used to score the effectiveness of any arbitrary visualization. Most importantly, it even enables us to score visualizations generated from entirely new datasets not used for training  $\mathcal{M}$ , i.e., a dataset  $X_{\text{test}}$  outside the training corpus  $X_{\text{test}} \notin \mathcal{D}$ .

$$\mathcal{M} : \mathcal{X}_{\text{test}} \times \mathcal{C} \rightarrow \mathbb{R} \quad (12)$$

Hence, given an arbitrary visualization,  $\mathcal{M}$  outputs a score describing the effectiveness or importance of the visualization. The ML-based model learning formulation for visualization recommendation shown in Eq. 11 can naturally be used to recover many different types of visualization recommendation models.

<sup>10</sup>Note  $\Psi$  and  $f$  can also be learned along with the model  $\mathcal{M}$  or learned/defined prior to learning the model  $\mathcal{M}$ .

**DEFINITION 8 (META-FEATURE FUNCTION).** Let  $\Psi$  denote the meta-feature learning function that maps an attribute  $\mathbf{x}$  of any dimensionality (from any dataset  $\mathbf{X}$ ) to a shared  $K$ -dimensional meta-feature space that captures the important characteristics of  $\mathbf{x}$ . More formally,

$$\Psi : \mathbf{x} \rightarrow \mathbb{R}^K \quad (13)$$

where  $\mathbf{x}$  can be of an arbitrary attribute type (e.g., real-valued, integral, nominal, ordinal, etc) and size, e.g., two attributes  $\mathbf{x} \in \mathbf{X}$  and  $\mathbf{y} \in \mathbf{Y}$  from two different datasets are almost surely of different dimensionality (# rows). Further, given  $M$  attributes of a dataset  $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$ , then

$$\Psi : \mathbf{X} \rightarrow \mathbb{R}^{K \times M} \quad (14)$$

Hence, from Eq. 13  $\Psi(\mathbf{x}) \in \mathbb{R}^K$  and  $\Psi(\mathbf{X}) \in \mathbb{R}^{K \times M}$ .

### 3.2 Recommending Visualizations via Model

Once we have learned the visualization recommendation model  $\mathcal{M}$  (Eq. 11) using the training visualization corpus  $\mathcal{D}$ , then we can use  $\mathcal{M}$  to score and recommend a list of the top most important and insightful visualizations generated from an arbitrary new dataset  $\mathbf{X}_{\text{test}} \notin \mathcal{D}$ .

**DEFINITION 9 (ML-BASED VISUALIZATION RECOMMENDATION).** Let  $\mathcal{M}$  be the trained visualization recommender model from Def. 7. Given  $\mathcal{M}$  along with a new (unseen) dataset  $\mathbf{X}_{\text{test}} \notin \{\mathbf{X}_i\}_{i=1}^N$  of interest, then

$$\mathcal{M} : \mathcal{X}_{\text{test}} \times \mathcal{C} \rightarrow \mathbb{R} \quad (15)$$

where  $\mathcal{X}_{\text{test}} = \{\dots, \mathbf{X}_{\text{test}}^{(k)}, \dots\}$  is the space of attribute combinations from  $\mathbf{X}_{\text{test}}$  and  $\mathcal{C}$  is the space of visualization configuration. Given the set of generated visualizations  $\mathbb{V}_{\text{test}} = \{\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_Q\}$ , we derive a ranking of the visualizations  $\mathbb{V}_{\text{test}}$  from  $\mathbf{X}_{\text{test}}$  as follows:

$$\rho(\{\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_Q\}) = \arg \operatorname{sort}_{\mathcal{V}_t \in \mathbb{V}_{\text{test}}} \mathcal{M}(\mathcal{V}_t) \quad (16)$$

where  $Q = |\mathbb{V}_{\text{test}}|$ . Hence, given an arbitrary visualization,  $\mathcal{M}$  outputs a score describing the effectiveness or importance of the visualization.

Informally, given a new dataset  $\mathbf{X}_{\text{test}}$  to recommend visualizations for via the trained model  $\mathcal{M}$  (Eq. 11), then  $\mathcal{M}(\xi(\Sigma(\mathbf{X}_{\text{test}}), \mathcal{C}))$  where  $\mathcal{C}$  is the space of relevant visualization configurations. Notice that  $\mathcal{M}(\mathbb{V}_{\text{test}}) = \mathcal{M}(\xi(\Sigma(\mathbf{X}_{\text{test}}), \mathcal{C}))$ . For tractability, we replace the set of possible visualization configurations  $\mathcal{C}$  with the set of relevant configurations  $\mathcal{C}_r = \mathcal{R}(\mathcal{C})$  where  $\mathcal{R}$  is a function consisting of visual rules that enables us to discard configurations that are invalid with respect to the manually defined rules. The list of rules from Voyager and other rule-based systems can read from a file similar to stopwords in information retrieval. Hence,  $\mathcal{C}_r \subseteq \mathcal{C}$ .

Given a new dataset of interest, the space of visualizations to search over is completely different from the space of visualizations that arises from any other (non-identical) dataset. More formally, let  $\mathbb{V}_i^*$  and  $\mathbb{V}_j^*$  denote the space of all possible visualizations that arise from  $\mathbf{X}_i$  and  $\mathbf{X}_j$  held-out datasets, then  $\forall r, s, \mathcal{V}_r \in \mathbb{V}_i^* \neq \mathcal{V}_s \in \mathbb{V}_j^*$  holds. Further, this obviously holds  $\forall i, j \in [T]$  as well. Clearly, the above holds, since a visualization consists of a subset of attributes (data) and design choices. The above demonstrates the difficulty of the visualization recommendation learning problem, in the sense that, the model must recommend relevant visualizations

from a space of visualizations never seen by the learning algorithm. Moreover, we can even show a weaker property regarding the cardinality of the space of visualizations that arise from different held-out datasets,

**CLAIM 1.** Let  $\mathbb{V}_i^*$  and  $\mathbb{V}_j^*$  denote the space of all possible visualizations that arise from  $\mathbf{X}_i$  and  $\mathbf{X}_j$  held-out datasets, then with high probability  $|\mathbb{V}_i^*| \neq |\mathbb{V}_j^*|$  almost surely holds  $\forall i, j \in [T]$ .

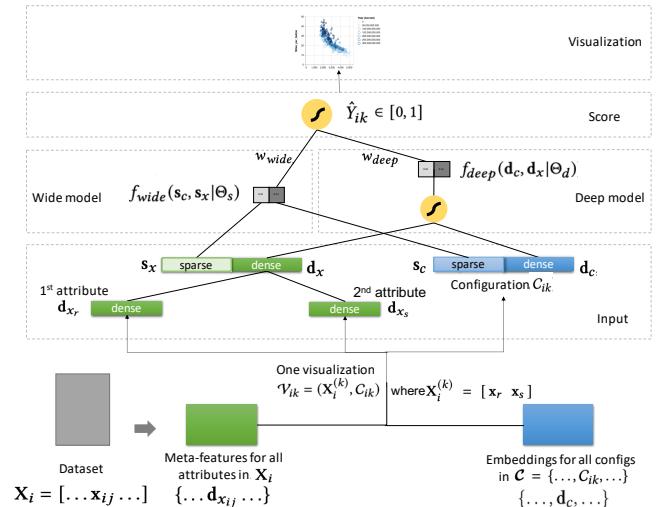
## 4 WIDE & DEEP VISUALIZATION RECOMMENDATION

Following the general ML-based visualization recommendation formulation in Section 3, we now describe our proposed wide-and-deep visualization recommendation approach. Table 1 provides a summary of the key notation.

### 4.1 Wide-and-Deep Network Overview

We now give a brief overview of the wide-and-deep learning-based visualization recommendation approach. Figure 2 shows the wide-and-deep network architecture.

- **Encoding Visualizations and Their Attributes (Sec. 4.2):** The network first encodes the visualization  $\mathcal{V}_{ik}$  from one arbitrary dataset  $\mathbf{X}_i \in \mathcal{D}$  by its attribute combination  $\mathbf{X}_i^{(k)}$  and the visualization configuration  $C_{ik}$  into dense and sparse features (Section 4.2), denoted as  $\mathbf{d}_x$ ,  $\mathbf{d}_c$ ,  $\mathbf{s}_x$  and  $\mathbf{s}_c$ .
- **Wide Vis. Rec. Model (Sec. 4.3):** The wide model takes as input the sparse features  $\mathbf{s}_x$  and  $\mathbf{s}_c$  of  $\mathbf{X}_i^{(k)}$  and  $C_{ik}$ , then outputs a wide score  $f_{\text{wide}}(\mathbf{s}_c, \mathbf{s}_x | \Theta_s)$ . The wide model uses a linear model over



**Figure 2: For an arbitrary dataset  $\mathbf{X}_i$  (either a new unseen dataset or one from the training corpus), we generate the space of visualizations  $\mathbb{V}_i^*$  for  $\mathbf{X}_i$ . The visualizations then feed our wide-and-deep network model  $\mathcal{M}$  one-by-one. For each visualization, the network takes as input the attribute combination  $\mathbf{X}_i^{(k)}$  and a configuration  $C_{ik}$ , and outputs a score  $\hat{Y}_{ik}$  as the predicted effectiveness of this visualization.**

cross-product feature transformations to capture any occurrence of feature-pairs that commonly leads to effective visualizations.

- **Deep Vis. Rec. Model (Sec. 4.4):** The *deep* model takes as input the dense features  $\mathbf{d}_x$  and  $\mathbf{d}_c$  of  $\mathbf{X}_i^{(k)}$  and  $C_{ik}$ , then outputs a deep score  $f_{\text{deep}}(\mathbf{d}_c, \mathbf{d}_x | \Theta_d)$ . The *deep* model uses dense features and non-linear transformations to generalize to unseen feature pairs that do not appear in the training set yet may lead to effective visualizations.

• **Training (Sec. 4.5):** We describe the end-to-end *training* of the wide-and-deep network model  $\mathcal{M}$ . The model and its parameters are learned using SGD over a sample of training visualizations from the corpus  $\mathcal{D} = \{\mathbf{X}_i, \mathbb{V}_i\}_{i=1}^N$ .

• **Scoring & Recommending Visualizations via  $\mathcal{M}$  (Sec. 4.6):** Given an entirely new unseen dataset  $\mathbf{X}_{\text{test}}$ , we then describe the inference procedure that uses  $\mathcal{M}$  to score and recommend visualizations for the new dataset  $\mathbf{X}_{\text{test}}$  of interest.

Figure 2 illustrates how  $\mathcal{M}$  operates at the granular level during both training (Sec. 4.5) and inference (Sec. 4.6): it predicts a numerical score  $\hat{Y}_{ik}$  for each visualization  $\mathcal{V}_{ik} = (\mathbf{X}_i^{(k)}, C_{ik})$  of a specific dataset  $\mathbf{X}_i$ <sup>11</sup>. The score  $\hat{Y}_{ik}$  is given by

$$\hat{Y}_{ik} = \mathcal{M}(\mathcal{V}_{ik}) = f(\mathbf{X}_i^{(k)}, C_{ik} | \Theta) \in [0, 1] \quad (17)$$

## 4.2 Encoding the Input

Every visualization can naturally be decomposed into the subset of attributes from the dataset  $\mathbf{X}_i$  and the visualization configuration, i.e.,  $\mathcal{V}_{ik} = (\mathbf{X}_i^{(k)}, C_{ik})$ . Since both  $\mathcal{V}_{ik}$  and  $\mathbf{X}_i^{(k)}$  are specific to an arbitrary dataset  $\mathbf{X}_i$ , the first step is to encode the input  $\mathbf{X}_i^{(k)}$  and  $C_{ik}$  into features in some shared space for the network.

**4.2.1 Encode attributes into meta-features.** Attributes from datasets in the training corpus  $\{\mathbf{X}_i\}_{i=1}^N$  are naturally from different domains and have fundamentally different characteristics such as their types, sizes, meanings, and so on. This makes it fundamentally important to encode every attribute from any dataset in the corpus in a shared  $K$ -dimensional space where we can naturally characterize similarity between the attributes. For this purpose, we leverage the meta-feature function  $\Psi$  from Def. 8. Since  $\Psi$  represents an attribute  $\mathbf{x}$  in a shared  $K$ -dimensional space, we apply  $\Psi \forall \mathbf{x} \in \mathbf{X}_i$ .

We propose the meta-feature learning framework, as an instance of  $\Psi$  for the network. The framework has several components including nested meta-feature functions, attribute representation functions (where each of which can be used with the set of nested meta-feature functions), and so on. The framework is summarized in Table 2. More formally, first we compute the meta-feature functions over different representations of the data as follows:

$$\psi(\mathbf{x}), \psi(p(\mathbf{x})), \psi(g(\mathbf{x})), \dots, \quad (18)$$

where  $\psi(\mathbf{x})$  is the meta-features from  $\mathbf{x}$  directly,  $\psi(p(\mathbf{x}))$  are the meta-features from the probability distribution of  $\mathbf{x}$ , and so on. Next, given a partitioning (or clustering, binning) function  $\Pi$  that divides a vector  $\mathbf{x}$  (or  $p(\mathbf{x}), g(\mathbf{x})$ ) of values into  $k$  partitions, we can

<sup>11</sup> $\mathbf{X}_i$  can be either a new unseen dataset  $\mathbf{X}_{\text{test}}$ , or a dataset from the training corpus  $\mathcal{D} = \{\mathbf{X}_i, \mathbb{V}_i\}_{i=1}^N$

**Table 1: Summary of notation. Matrices are bold upright roman letters; vectors are bold lowercase letters.**

$\mathcal{D} = \{\mathbf{X}_i, \mathbb{V}_i\}_{i=1}^N$	a corpus of datasets $\{\mathbf{X}_i\}_{i=1}^N$ and $N$ sets of visualizations $\{\mathbb{V}_i\}_{i=1}^N$
$\mathbf{X}_i = [\dots \mathbf{x}_{ij} \dots]$	an arbitrary dataset that has many attributes
$\mathbf{x}$	an attribute vector from an arbitrary dataset
$\mathbb{V}_i^\star = \mathbb{V}_i^- \cup \mathbb{V}_i^+$	space of all possible visualizations that can be generated from $\mathbf{X}_i$ , also written as $\{\dots, \mathcal{V}_{ik}, \dots\}$
$\mathbb{V}_i \subseteq \mathbb{V}_i^\star$	set of positive visualizations (user-generated, observed) in $\mathbb{V}_i^\star$
$\mathbb{V}_i^- \subseteq \mathbb{V}_i^\star$	set of negative visualizations in $\mathbb{V}_i^\star$
$\hat{\mathbb{V}}_i^- \subseteq \mathbb{V}_i^-$	sampled negative visualizations from $\mathbb{V}_i^-$
$\mathcal{X}_i = \{\dots \mathbf{X}_i^{(k)}, \dots\}$	space of attribute combinations for dataset $\mathbf{X}_i$
$\mathcal{V}_{ik} = (\mathbf{X}_i^{(k)}, C_{ik})$	a visualization $\mathcal{V}_{ik}$ of dataset $\mathbf{X}_i$ consisting of the attributes used in the visualization $\mathbf{X}_i^{(k)}$ and a visualization configuration $C_{ik}$
$\mathbf{X}_i^{(k)} \subseteq \mathbf{X}_i$	attribute combination in a visualization $\mathcal{V}_{ik}$
$\mathcal{C} = \{\dots, C_{ik}, \dots\}$	space of all visualization configurations
$C_{ik} \in \mathcal{C}$	a visualization configuration
$\mathcal{M}$	a visualization recommendation model
$f(\mathbf{X}_i^{(k)}, C_{ik}   \Theta) = \hat{Y}_{ik}$	scoring function of $\mathcal{M}$ , parameterized by $\Theta$ on the input of $\mathbf{X}_i^{(k)}$ and $C_{ik}$
$g(\cdot)$	vector normalization
$\phi_1$	a concatenation operator
$\phi_k(\mathbf{s})$	$k$ -th cross-product transformation function
$\Theta$	entire set of model parameters in $\mathcal{M}$
$\mathbf{W}_s^T$	weight matrix of the wide model
$\mathbf{b}$	bias vector of the wide model
$\Theta_s = \{\mathbf{W}_s^T, \mathbf{b}\}$	model parameters for the wide model
$\{\mathbf{W}_{d1}, \dots, \mathbf{W}_{dL}\}$	weight matrices of the deep model
$\{\mathbf{b}_{d1}, \dots, \mathbf{b}_{dL}\}$	bias vectors of the deep model
$\Theta_d = \{\mathbf{W}_{d1}, \dots, \mathbf{b}_{d1}, \dots\}$	model parameters for the deep model
$w_{\text{wide}}$	weight to combine the wide score $f_{\text{wide}}(\mathbf{s}_c, \mathbf{s}_x   \Theta_s)$
$w_{\text{deep}}$	weight to combine the deep score $f_{\text{deep}}(\mathbf{d}_c, \mathbf{d}_x   \Theta_d)$
$\mathbf{s}_x$	sparse feature vector for an attribute combination $\mathbf{X}_i^{(k)}$
$\mathbf{s}_c$	sparse feature vector for a configuration $C_{ik}$
$\mathbf{s} = \phi_1(\mathbf{s}_c, \mathbf{s}_x)$	concatenated sparse features
$\mathbf{s}'$	cross-product feature vector
$\mathbf{d}_x$	dense feature vector for attribute combination $\mathbf{X}_i^{(k)}$
$\mathbf{d}_c$	dense feature vector for a visualization configuration $C_{ik}$
$\mathbf{d} = \phi_1(\mathbf{d}_c, \mathbf{d}_x)$	concatenated dense features

derive meta-features for each partition as follows:

$$\psi(\Pi_1(\mathbf{x})), \dots, \psi(\Pi_k(\mathbf{x})), \quad (19)$$

$$\psi(\Pi_1(p(\mathbf{x}))), \dots, \psi(\Pi_k(p(\mathbf{x}))), \quad (20)$$

$$\psi(\Pi_1(g(\mathbf{x}))), \dots, \psi(\Pi_k(g(\mathbf{x}))) \quad (21)$$

In the above, we use  $\Pi_k$  to denote the  $k$ th partition of values from the partitioning function  $\Pi$ . In this work, leverage multiple partitioning functions, and each can be used in a similar fashion as

**Table 2: Meta-feature framework for an attribute  $\mathbf{x}$ .**

FRAMEWORK COMPONENTS	EXAMPLES
<b>1. Attribute representations</b>	$\mathbf{x}, p(\mathbf{x}), g(\mathbf{x}), \ell_B(\mathbf{x}), \dots$
<b>2. Partitioning values II</b>	Clustering, binning, quartiles, ...
<b>3. Meta-feature functions <math>\psi</math></b>	Statistical, info theoretic, ...

shown in Eq. 19. All the meta-features derived from Eq. 18 and Eq. 19 are then concatenated into a single vector of meta-features describing the characteristics of the attribute  $\mathbf{x}$ . More formally, the meta-feature function  $\Psi : \mathbf{x} \rightarrow \mathbb{R}^K$  is defined as follows:

$$\Psi(\mathbf{x}) = [\psi(\mathbf{x}), \psi(p(\mathbf{x})), \psi(g(\mathbf{x})), \dots, \psi(\Pi_1(\mathbf{x})), \dots, \psi(\Pi_k(\mathbf{x})), \dots, \psi(\Pi_1(p(\mathbf{x}))), \dots, \psi(\Pi_k(p(\mathbf{x}))), \dots, \psi(\Pi_1(g(\mathbf{x}))), \dots, \psi(\Pi_k(g(\mathbf{x})))] \quad (22)$$

As an example, given an attribute vector  $\mathbf{x}$  from any arbitrary dataset  $X$ , the first step is to derive many different data representations of  $\mathbf{x}$ , e.g., using different normalization/scaling functions, probability distribution, log binning of  $\mathbf{x}$ , etc. Then, we partition the values of each of the different representations of  $\mathbf{x}$  previously computed in Step 1 of Table 2. Now, for every different data representation of  $\mathbf{x}$  from Step 1 and every different partition of values from Step 2, we apply meta-feature functions from Step 3 (see Table 3) over each one to get meta-features of  $\mathbf{x}$ . Finally, we concatenate the meta-features from Step 3. The resulting  $\Psi(\mathbf{x})$  is a dense vector. We denote it as  $\mathbf{d}_x$  - the *meta-features*, a.k.a. the dense feature of the attribute  $\mathbf{x}$ . Without loss generality, we also normalize each meta-feature in  $\mathbf{d}_x$ , by min-max scaling to scale each meta-feature value in  $\mathbf{d}_x$  to be between 0 and 1. Our approach is agnostic to the precise meta-feature functions used, and is flexible to use with any alternative set of meta-feature functions.

We obtain  $\Psi(\mathbf{x})$  as the meta-features for each attribute  $\mathbf{x}$ . An attribute combination  $X_i^{(k)}$  usually has more than one attributes, whose meta-features need to be combined to get an overall dense feature  $\mathbf{d}_x$ . We concatenate all of meta-features  $\mathbf{d}_{x_{ij}}$  from each attribute  $\mathbf{x}_{ij} \in X_i^{(k)}$  to get the overall dense feature  $\mathbf{d}_x$ , written as

$$\mathbf{d}_x = \phi_1(\dots \mathbf{d}_{x_{ij}} \dots) = \begin{bmatrix} \vdots \\ \mathbf{d}_{x_{ij}} \\ \vdots \end{bmatrix} \quad (23)$$

See Figure 2 for an example. It has two attributes selected, i.e.  $X_i^{(k)} = [\mathbf{x}_r \ \mathbf{x}_s]$ , which results in two dense vectors  $\mathbf{d}_{x_r}$  and  $\mathbf{d}_{x_s}$ . The overall dense feature  $\mathbf{d}_x$  therefore is  $\mathbf{d}_x = \phi_1(\mathbf{d}_{x_r}, \mathbf{d}_{x_s})$ .

**4.2.2 Visualization Configuration Embedding.** The space of all possible configurations  $\mathcal{C} = \{\dots, C_{ik}, \dots\}$  is a shared set for all visualizations from any dataset. Let  $C_{ik}$  denote one configuration in the space. Like all other configurations, although we denote  $C_{ik}$  as the configuration of the visualization of our interest, where  $V_{ik} = (X_i^{(k)}, C_{ik})$ , this configuration is independent from any dataset or visualization (Property 2). It is possible to learn embeddings for all configurations in  $\mathcal{C}$  and use the embeddings to encode  $C_{ik}$ .

**DEFINITION 10 (CONFIGURATION EMBEDDING FUNCTION).** Let  $E$  denote a configuration embedding function that maps a configuration  $C_{ik}$  to a shared  $K$ -dimensional embedding space such that the

**Table 3: Summary of meta-feature functions for attribute.** The functions will be called from the learning framework in Table 2. Let  $\mathbf{x}$  denote an arbitrary attribute vector and  $\pi(\mathbf{x})$  is the sorted vector of  $\mathbf{x}$ .

Name	Equation
Num. instances	$ \mathbf{x} $
Num. missing values	$s$
Frac. of missing values	$ \mathbf{x}  - s /  \mathbf{x} $
Num. nonzeros	$\text{nnz}(\mathbf{x})$
Num. unique values	$\text{card}(\mathbf{x})$
Density	$\text{nnz}(\mathbf{x}) /  \mathbf{x} $
$Q_1, Q_3$	median of the $ \mathbf{x}  / 2$ smallest (largest) values
IQR	$Q_3 - Q_1$
Outlier LB $\alpha \in \{1.5, 3\}$	$\sum_i \mathbb{I}(x_i < Q_1 - \alpha IQR)$
Outlier UB $\alpha \in \{1.5, 3\}$	$\sum_i \mathbb{I}(x_i > Q_3 + \alpha IQR)$
Total outliers $\alpha \in \{1.5, 3\}$	$\sum_i \mathbb{I}(x_i < Q_1 - \alpha IQR) + \sum_i \mathbb{I}(x_i > Q_3 + \alpha IQR)$
( $\alpha$ std) outliers $\alpha \in \{2, 3\}$	$\mu_x \pm \alpha \sigma_x$
Spearman ( $\rho$ , p-val)	$\text{spearman}(\mathbf{x}, \pi(\mathbf{x}))$
Kendall ( $\tau$ , p-val)	$\text{kendall}(\mathbf{x}, \pi(\mathbf{x}))$
Pearson ( $r$ , p-val)	$\text{pearson}(\mathbf{x}, \pi(\mathbf{x}))$
Min, max	$\min(\mathbf{x}), \max(\mathbf{x})$
Range	$\max(\mathbf{x}) - \min(\mathbf{x})$
Median	$\text{med}(\mathbf{x})$
Geometric Mean	$ \mathbf{x} ^{-1} \prod_i x_i$
Harmonic Mean	$ \mathbf{x}  / \sum_i \frac{1}{x_i}$
Mean, Stdev, Variance	$\mu_x, \sigma_x, \sigma_x^2$
Skewness	$\mathbb{E}(x - \mu_x)^3 / \sigma_x^3$
Kurtosis	$\mathbb{E}(x - \mu_x)^4 / \sigma_x^4$
HyperSkewness	$\mathbb{E}(x - \mu_x)^5 / \sigma_x^5$
Moments [6-10]	—
k-statistic [3-4]	—
Quartile Dispersion Coeff.	$\frac{Q_3 - Q_1}{Q_3 + Q_1}$
Median Absolute Deviation	$\text{med}( \mathbf{x} - \text{med}(\mathbf{x}) )$
Avg. Absolute Deviation	$\frac{1}{ \mathbf{x} } \mathbf{x}^T  \mathbf{x} - \mu_x $
Coeff. of Variation	$\sigma_x / \mu_x$
Efficiency ratio	$\sigma_x^2 / \mu_x^2$
Variance-to-mean ratio	$\sigma_x^2 / \mu_x$
Signal-to-noise ratio (SNR)	$\mu_x^2 / \sigma_x^2$
Entropy	$H(\mathbf{x}) = - \sum_i x_i \log x_i$
Norm. entropy	$H(\mathbf{x}) / \log_2  \mathbf{x} $
Gini coefficient	—
Quartile max gap	$\max(Q_{i+1} - Q_i)$
Centroid max gap	$\max_{ij}  c_i - c_j $
Histogram prob. dist.	$p_h = \frac{h}{h^T \mathbf{e}}$ (with fixed # of bins)

embedding  $\mathcal{H}(\mathcal{C})$  captures the important characteristics of  $C_{ik}$  and can be learned along with the model  $\mathcal{M}$ . More formally,

$$\mathcal{H} : C_{ik} \rightarrow \mathbb{R}^K \quad (24)$$

Further, given the space of all visualization configurations  $\mathcal{C}$  of size  $M = |\mathcal{C}|$ , then we obtain a  $K$ -dimensional embedding matrix for all visualization configurations as  $\mathcal{H}(\mathcal{C})$

$$\mathcal{H} : \mathcal{C} \rightarrow \mathbb{R}^{K \times M} \quad (25)$$

We denote  $\mathcal{H}(C_{ik})$  as the dense feature  $\mathbf{d}_c$  of the configuration  $C_{ik}$ , i.e.  $\mathbf{d}_c = \mathcal{H}(C_{ik})$ .  $\mathcal{H}$  works as follows: Suppose we are scoring

visualizations for an arbitrary dataset, one visualization is  $\mathcal{V}_{ik} = (\mathbf{X}_i^{(k)}, C_{ik})$ . We first abstract the configuration  $C_{ik}$  from  $\mathcal{V}_{ik}$ , and look up the positional identity of  $C_{ik}$  in  $\mathcal{C}$ . Then, we one-hot encode the identity  $C_{ik}$  and apply configuration embedding function  $\mathcal{H}$  to the one-hot encoding. This gives a  $k$ -dimensional dense feature  $\mathbf{d}_c$ , written as

$$\mathbf{d}_c = \mathcal{H}(\text{one\_hot}(C_{ik})) \quad (26)$$

Note that  $\mathcal{H}$  is learnable with the model  $\mathcal{M}$ .

**4.2.3 Complement Dense Features with Sparse Features.** Up so far, both the configuration embedding vector  $\mathbf{d}_c$  and attribute meta-features  $\mathbf{d}_x$  are dense features (vectors in real-value). On the other hand, our approach wants to capture some frequent feature patterns about the attribute combination  $\mathbf{X}_i^{(k)}$  and the configuration  $C_{ik}$  that commonly lead to effective visualizations. The frequent feature patterns can be best expressed through sparse features, i.e. *whether this visualization has the feature(s) X or not*. For example, scatterplot is generally more effective to visualize attributes that have many rows, than line charts and bar charts. If a visualization  $\mathcal{V}_i^{(k)} = (\mathbf{X}_i^{(k)}, C_{ik})$  has sparse features indicating that the number of rows in one attribute of  $\mathbf{X}_i^{(k)}$  is larger than 50 and the configuration  $C_{ik}$  is about scatterplot, our model  $\mathcal{M}$  should be able to assign a high score to this visualization and consider it as effective. Therefore, we create the set of sparse features  $\mathbf{s}_x$  and  $\mathbf{s}_c$  to complement the dense features  $\mathbf{d}_x$  and  $\mathbf{d}_c$ , which will also be used as the input to  $\mathcal{M}$ .

There are many choices to create sparse features  $\mathbf{s}_x$  and  $\mathbf{s}_c$ . For example, one simple option to get the sparse feature  $\mathbf{s}_x$  for attribute combination  $\mathbf{X}_i^{(k)}$  is to bin-bucket the dense features  $\mathbf{d}_x$ . Recall that the dense feature (i.e. meta-features)  $\mathbf{d}_x$  is a vector normalized in each dimension. We could bin-bucket each dimension of the normalized meta-features  $\mathbf{d}_x$  into a fixed number of  $n$ -bins within the range of  $[0, 1]$ . Each bin has an equal width of  $\frac{1}{n}$ . Another option to get the sparse features from  $\mathbf{d}_x$  is to first cluster each dimension from  $\mathbf{d}_x$  of all seen visualizations, then one-hot encode the cluster identity for the value in each dimension of the dense feature  $\mathbf{d}_x$ . Our wide-and-deep network is agnostic to the actual option and the precise meta-features that are being used in  $\mathbf{d}_x$ .

To get the sparse feature  $\mathbf{s}_c$  from a configuration embedding vector  $\mathbf{d}_c$ , one option is to use the original one-hot sparse vector as its sparse feature. Another option is to one-hot encode each pair of field and value that appears in the configuration  $C_{ik}$ . For example, we could assign a value of 1 to one dimension of  $\mathbf{s}_c$  for a configuration  $C_{ik}$ , if the configuration  $C_{ik}$  satisfies a specific pair of field and value, such as “marker.symbol = circle.”

### 4.3 The Wide Model

The *Wide* model is a linear model over the set of sparse features  $\mathbf{s}_c$  and  $\mathbf{s}_x$ . The goal of leveraging sparse features is to capture any occurrence of feature-pairs that commonly lead to effective visualizations in the training corpus. As an example, if the corpus  $\mathcal{D} = \{\mathbf{X}_i, \mathbb{V}_i\}_{i=1}^N$  has many visualizations that use scatterplot with default point size and point color to visualize two quantitative

attributes with more than 50 rows,<sup>12</sup> a fully-trained model  $\mathcal{M}$  should be able to pick up this pattern: when a new dataset comes in, which has over 50 rows and at least two quantitative attributes in similar characteristics,  $\mathcal{M}$  would be able to generate, score, and recommend a similar-style scatterplot that visualizes over a subset of two quantitative attributes.

First, we concatenate  $\mathbf{s}_c$  and  $\mathbf{s}_x$  into one single sparse vector  $\mathbf{s}$  where  $\phi_1$  is a concatenation operator.

$$\mathbf{s} = \phi_1(\mathbf{s}_c, \mathbf{s}_x) = \begin{bmatrix} \mathbf{s}_c \\ \mathbf{s}_x \end{bmatrix} \quad (27)$$

Next, we augment  $\mathbf{s}$  with *cross-product features* from  $\mathbf{s}$ , denoted as  $\mathbf{s}'$ . *Cross-product features*  $\mathbf{s}'$  captures co-occurrences of some specific features in the original  $\mathbf{s}$ . Formally, it is calculated as the concatenation of values from a set of cross-product transformation functions.

$$\mathbf{s}' = \{\dots, \phi_k(\mathbf{s}), \dots\} \quad (28)$$

where  $\phi_k(\mathbf{s})$  is the  $k$ -th cross-product transformation function. The operator  $\phi_k(\cdot)$  checks whether a few selected dimensions in  $\mathbf{s}$  are all 1, written as

$$\phi_k(\mathbf{s}) = \prod_{i=1}^{|\mathbf{s}|} \mathbf{s}_i^{t_{ki}}, \quad t_{ki} \in \{0, 1\} \quad (29)$$

where  $t_{ki}$  is a boolean value indicating whether or not the  $k$ -th cross-product transformation function  $\phi_k(\mathbf{s})$  “cares” about the  $i$ -th feature of  $\mathbf{s}$ . For example, suppose  $\phi_k(\cdot)$  checks whether a visualization satisfies (1) the entropy of its first attribute is in the range of  $[0.2, 0.4]$  and (2) its configuration is configuration no.3. The cross-product feature  $\phi_k(\mathbf{s})$  is 1 if and only  $\mathbf{s}$  has feature dimensions of `entropy-1st-var-bucket=2` and `config-bucket=3` both as 1.

Finally, the sparse feature  $\mathbf{s}$  and the cross-product transformed feature  $\mathbf{s}'$  get concatenated using the concatenation operator  $\phi_1$ , then go through a linear transformation, to get the wide score. More formally, the wide score is

$$f_{\text{wide}}(\mathbf{s}_c, \mathbf{s}_x | \Theta_s) = \mathbf{W}_s^T [\mathbf{s}, \mathbf{s}'] + \mathbf{b}_s \quad (30)$$

where  $\mathbf{W}_s^T$  and  $\mathbf{b}$  denote the weight matrix and the bias vector for the wide model.  $\Theta_s = \{\mathbf{W}_s^T, \mathbf{b}\}$  denotes the entire set of parameters in the wide model. The wide score is a numeric value, i.e. satisfies  $f_{\text{wide}}(\mathbf{s}_c, \mathbf{s}_x | \Theta_s) \in \mathbb{R}$ .

### 4.4 The Deep Model

The *Deep* model uses dense features and non-linear transformations to generalize to feature pairs that do not frequently appear in the training set yet may lead to effective visualizations with a good rationale. Suppose the same corpus  $\mathcal{D} = \{\mathbf{X}_i, \mathbb{V}_i\}_{i=1}^N$  as in Sec. 4.3 not only has the frequently-observed pattern about scatterplots, but also a few scatterplots with half point size that visualize two quantitative attributes with hundreds of rows, a fully-trained model  $\mathcal{M}$  should be able to generalize from this. When a new dataset with thousands of rows and at least two quantitative attributes comes in,  $\mathcal{M}$  would be able to generate, score, and recommend a scatterplot that preferably has smaller point size to visualize a subset of two quantitative attributes.

<sup>12</sup>Note on this pattern, which will be reused in Sec. 4.4 for motivations of the *Deep* model.

The deep model works by first concatenating the two dense features  $\mathbf{d}_c$  and  $\mathbf{d}_x$  into an intermediate vector  $\mathbf{d}$ , such that it incorporates the information from both the configuration and the attribute combination.

$$\mathbf{d} = \phi_1(\mathbf{d}_c, \mathbf{d}_x) = \begin{bmatrix} \mathbf{d}_c \\ \mathbf{d}_x \end{bmatrix} \quad (31)$$

The concatenated vector  $\mathbf{d}$  are then fed into a total of  $L$  hidden layers (standard MLP layers). The initial layer starts with  $\mathbf{d}$ . At the  $k$ -th layer, an intermediate vector  $\mathbf{d}_{k-1}$  from the previous layer ( $k-1$ ) go through non-linear transformations with the model parameter  $\mathbf{W}_{dk}$  and the activation function  $a_{k-1}$ . The activation function  $a_{k-1}$  could either be the rectified linear unit (ReLU) or the sigmoid function. This design offers greater flexibility to model feature interaction. The last layer gives the output from the deep model, as the deep score  $f_{deep}(\mathbf{d}_c, \mathbf{d}_x | \Theta_d)$ . Formally, it can be written as

$$\begin{aligned} \mathbf{d}_0 &= \mathbf{d} \\ \mathbf{d}_1 &= a_1(\mathbf{W}_{d1}^T \mathbf{d}_0 + \mathbf{b}_{d1}), \\ &\dots \\ \mathbf{d}_{L-1} &= a_{L-1}(\mathbf{W}_{d(L-1)}^T \mathbf{d}_{L-2} + \mathbf{b}_{d(L-1)}), \\ f_{deep}(\mathbf{d}_c, \mathbf{d}_x | \Theta_d) &= a_L(\mathbf{W}_{dL}^T \mathbf{d}_{L-1} + \mathbf{b}_{dL}), \end{aligned} \quad (32)$$

where  $\{\mathbf{W}_{d1}, \dots, \mathbf{W}_{dL}\}$  and  $\{\mathbf{b}_{d1}, \dots, \mathbf{b}_{dL}\}$  denote the weight matrices and the bias vectors for the deep model, and  $\Theta_d = \{\mathbf{W}_{d1}, \dots, \mathbf{W}_{dL}^T, \mathbf{b}_{d1}, \dots, \mathbf{b}_{dL}\}$  denotes the entire set of parameters in the deep model. The deep score is a numeric value, i.e. satisfies  $f_{deep}(\mathbf{d}_c, \mathbf{d}_x | \Theta_d) \in \mathbb{R}$ .

## 4.5 Training the Network

Previous two sections describe the set of model parameters  $\Theta$  that constitutes the wide-and-deep network and that goes into Eq. 17  $\hat{Y}_{ik} = \mathcal{M}(\mathcal{V}_{ik}) = f(\mathbf{X}_i^{(k)}, C_{ik} | \Theta)$ . In this section, we elaborate upon Def. 7 to show how to optimize the wide-and-deep network parameters  $\Theta$  with a probabilistic approach [2].

The training corpus  $\mathcal{D} = \{\mathbf{X}_i, \mathbb{V}_i\}_{i=1}^N$  has a set of datasets  $\{\mathbf{X}_i\}_{i=1}^N$ . Each dataset  $\mathbf{X}_i$  has a set of positive visualizations  $\mathbb{V}_i$ , which we also complement a sampled set of negative visualizations  $\hat{\mathbb{V}}_i^-$  (as in Def. 6). The set of training visualizations for  $\mathbf{X}_i$  is then  $\mathbb{V}_i \cup \hat{\mathbb{V}}_i^-$ . In other words, during training, each visualization  $\mathcal{V}_{ik}$  comes from an arbitrary dataset  $\mathbf{X}_i$  and has a binary ground-truth label  $Y_{ik} \in \{0, 1\}$ . A label of 1 indicates a positive visualization, i.e.  $\mathcal{V}_{ik} \in \mathbb{V}_i$ . Hence, the visualization is generated by the user. Label 0 indicates a negative (non-relevant) visualization, i.e.  $\mathcal{V}_{ik} \in \hat{\mathbb{V}}_i^-$ . Non-relevant visualizations are sampled from the space of all visualizations that belong to the dataset  $\mathbf{X}_i$ . Def. 6 and Section 5.3 discuss more details about how we sample non-relevant visualizations to support training.

Our goal is to have the model score  $\hat{Y}_{ik} \in [0, 1]$  of each training visualization  $\mathcal{V}_{ik}$  as close as possible to its ground-truth label  $Y_{ik}$ . We train the model by optimizing the likelihood of model scores rounding up to match the ground-truth labels, for all visualizations throughout the entire corpus  $\mathcal{D}$  consisting of  $N$  datasets  $\{\mathbf{X}_i\}_{i=1}^N$ . Eq. 33 shows the calculation of the likelihood: for each dataset  $\mathbf{X}_i$

we have the set  $\mathbb{V}_i \cup \hat{\mathbb{V}}_i^- = \{\dots, (\mathbf{X}_i^{(k)}, C_{ik}), \dots\}$  of training visualizations where each visualization  $(\mathbf{X}_i^{(k)}, C_{ik}) \in \mathbb{V}_i \cup \hat{\mathbb{V}}_i^-$  consists of the configuration  $C_{ik} \in \mathcal{C}$  and the subset of attributes  $\mathbf{X}_i^{(k)}$  from the dataset  $\mathbf{X}_i$ .

$$p(\hat{\mathbb{V}}_i^-, \mathbb{V}_i | \Theta) = \prod_{(\mathbf{X}_i^{(k)}, C_{ik}) \in \mathbb{V}_i} \hat{Y}_{ik} \prod_{(\mathbf{X}_i^{(k)}, C_{ik}) \in \hat{\mathbb{V}}_i^-} (1 - \hat{Y}_{ik}), \text{ for } i = 1, \dots, N \quad (33)$$

The closer that  $\hat{Y}_{ik}$  is to the ground-truth label  $Y_{ik}$ , the better. Taking the negative log of the likelihood in Eq. 33 and summing over all datasets  $\{\mathbf{X}_i\}_{i=1}^N$  give us the loss  $L$ .

$$\begin{aligned} L &= \sum_{i=1}^N \left( - \sum_{(\mathbf{X}_i^{(k)}, C_{ik}) \in \mathbb{V}_i} \log \hat{Y}_{ik} - \sum_{(\mathbf{X}_i^{(k)}, C_{ik}) \in \hat{\mathbb{V}}_i^-} \log(1 - \hat{Y}_{ik}) \right) \\ &= - \sum_{i=1}^N \sum_{(\mathbf{X}_i^{(k)}, C_{ik}) \in \mathbb{V}_i \cup \hat{\mathbb{V}}_i^-} Y_{ik} \log \hat{Y}_{ik} + (1 - Y_{ik}) \log(1 - \hat{Y}_{ik}) \end{aligned} \quad (34)$$

We minimize the objective function through stochastic gradient descent (SGD) to update the model parameters  $\Theta$  in  $\mathcal{M}$ .

## 4.6 Inference

Given the *trained* wide-and-deep visualization recommendation model  $\mathcal{M}$  from Section 4.5, we now describe the inference procedure for scoring and recommending visualizations from an arbitrary new dataset of interest. Recall from Section 3 that the set of visualizations to recommend depends entirely on the dataset of interest, that is, the set of visualizations for one dataset is guaranteed to be completely disjoint for another dataset. As illustrated in the lower part of Figure 2, given an arbitrary dataset  $\mathbf{X}_{test}$  selected or uploaded by an arbitrary user, we generate the space of visualizations  $\mathbb{V}_{test}^* = \{\dots, \mathcal{V}_{test}^{(k)}, \dots\}$  through Def. 3 process, where each visualization  $\mathcal{V}_{test}^{(k)}$  consists of a subset of attributes  $\mathbf{X}_{test}^{(k)}$  from the dataset  $\mathbf{X}_{test}$  and a configuration  $C \in \mathcal{C}$ , i.e.  $\mathcal{V}_{test}^{(k)} = (\mathbf{X}_{test}^{(k)}, C)$ .

Each visualization  $\mathcal{V}_{test}^{(k)}$  will be fed into  $\mathcal{M}$  for scoring. First, we encode the configuration  $C$  into the sparse feature  $\mathbf{s}_c$  and the dense feature (configuration embedding)  $\mathbf{d}_c$ . Given the attribute combination  $\mathbf{X}_{test}^{(k)}$ , we derive the meta-feature  $\mathbf{d}_{x_i}$  for each attribute  $x_i \in \mathbf{X}_{test}^{(k)}$ . The meta-features get concatenated to get an overall dense feature on attribute combination, as  $\mathbf{d}_x$ . Bin-bucking  $\mathbf{d}_x$  gives the sparse feature on attribute selection, as  $\mathbf{s}_x$ . The features go through the network where the wide model in Sec. 4.3 and the deep model in Sec. 4.4 transform them into a wide score and a deep score, denoted as  $f_{wide}(\mathbf{s}_c, \mathbf{s}_x | \Theta_s)$  and  $f_{deep}(\mathbf{d}_c, \mathbf{d}_x | \Theta_d)$  respectively, where  $\Theta_s$  and  $\Theta_d$  are model parameters in the wide model and the deep model. The network then weighs the two score vectors with respective weights, denoted as  $w_{wide}$  and  $w_{deep}$ , to get a final score  $\hat{Y}_{test,k} \in [0, 1]$  as follows,

$$\begin{aligned} \hat{Y}_{test,k} &= f(\mathbf{X}_{test}^{(k)}, C | \Theta) \\ &= \sigma(w_{wide} f_{wide}(\mathbf{s}_c, \mathbf{s}_x | \Theta_s) + w_{deep} f_{deep}(\mathbf{d}_c, \mathbf{d}_x | \Theta_d)) \end{aligned} \quad (35)$$

where  $f_{wide}(\mathbf{s}_c, \mathbf{s}_x | \Theta_s)$  is the wide score and  $f_{deep}(\mathbf{d}_c, \mathbf{d}_x | \Theta_d)$  is the deep score.  $w_{wide}$  and  $w_{deep}$  are two real values, i.e.  $w_{wide} \in \mathbb{R}$

and  $w_{deep} \in \mathbb{R}$ . The entire set of parameters  $\Theta$ , including  $\Theta_s$ ,  $\Theta_d$ ,  $w_{wide}$  and  $w_{deep}$  are learned through backward propagation as described in Section 4.5.

We repeat the above to score all possible visualizations in  $V_{test}^*$ , and then recommend top visualizations based on the prediction scores. This process is consistent with Def. 9. We expand more details about the evaluation of this process in Section 5.4.

## 5 EVALUATION FRAMEWORK

One of the contributions in this work is the proposed evaluation framework for end-to-end *ML-based* visualization recommendation systems. This framework serves as a fundamental basis for systematically evaluating ML-based visualization recommender systems, including our own model and those that arise in the future. We first summarize the differences that make it infeasible to use traditional techniques for evaluation of ML-based visualization recommendation systems, which motivates the need for such a framework, and then discuss each component of the evaluation framework.

### 5.1 Motivation

Since the visualization recommendation problem is fundamentally different from the traditional recommendation problem (*i.e.*, recommending items to users), we are unable to leverage the same commonly used evaluation techniques. We summarize some of these fundamental differences below, which motivate the need for the proposed evaluation framework.

- **Visualization Complexity (Sec. 5.2):** While traditional recommender systems have a simple object to recommend such as an item, ML-based visualization recommendation models must learn from a far more complex visualization object consisting of a subset of attributes from an arbitrary dataset, and a set of design choices.
- **No Shared Recommendation Space (Sec. 5.2):** Visualizations recommended for one dataset cannot be recommended for another dataset. Hence, there is no shared space of visualizations for learning better recommender models.
- **Generate On-The-Fly (Sec. 5.3-5.4):** Set of visualizations to recommend are generated on-the-fly for a specific unseen dataset of interest, as opposed to already existing and being common to all users as is the case for traditional recommender systems. For instance, when a user uploads a new dataset, ML-based visualization recommender systems must generate relevant visualizations that are only applicable for the user-specific dataset of interest.
- **Dynamic & Dataset Dependent Vis. Space (Sec. 5.3-5.4):** Space of visualizations to score and recommend is dynamic, completely *dependent* on the individual dataset of interest, and exponential in the number of attributes and possible design choices.

### 5.2 Corpus: Datasets and Visualizations

In most traditional recommender systems that recommend items to users, there is a *single shared set of items* (*e.g.*, movies on Netflix, products on Amazon, etc). However, in visualization recommendation, there is not a shared set of visualizations to recommend to users, as it depends entirely on the dataset of interest. Hence, if we have  $N$  datasets, then there are  $N$  completely disjoint sets

of visualizations that can be recommended. However, in visualization recommendation, we begin with a general corpus consisting of datasets and relevant visualizations. Each dataset has a set of relevant visualizations that are exclusive to the dataset. Moreover, each visualization only uses a small subset of attributes from the dataset. There can be attributes in the dataset that are never used in a visualization. While the goal of traditional recommender systems is typically to recommend items (from a specific dataset) to users, in visualization recommendation, the goal is to learn a model to score and ultimately recommend visualizations that are generated for a specific unseen dataset. Therefore, the model learned in visualization recommender systems must be able to generalize for use in scoring visualizations generated from any unseen dataset in the future.

Every new dataset gives rise to an exponential amount of possible visualizations. This makes this recommendation problem extremely challenging. In addition to the exponential space of visualizations that one must search for just a single dataset, the visualization search space is also completely disjoint from the search space of another arbitrary dataset as shown in Lemma 1. Therefore, given an available corpus that consists of datasets and relevant visualizations, our framework first splits the corpus by datasets into various sets required for training, validation, and testing. For datasets in the testing set, Section 5.5 discusses how to apply evaluation metrics to a number of ranked lists, where each list has recommended visualizations that are tied to one test dataset.

To learn a ML-based visualization recommendation model within our framework, we can use any available corpus as long as it has a set of datasets and relevant user-created visualizations that use a subset of attributes from the datasets. Notably, the corpus can be visualizations and datasets from a variety of different sources, *e.g.*, they can be visualizations and datasets collected from the web or even a visual analytics platform such as Tableau.

### 5.3 Training from the Corpus

The next step is to create a training set from the corpus of datasets and visualizations. For example, the wide-and-deep network approach addresses the issue of the dynamic space of visualizations by creating visualizations that come from a combination of attributes and a visualization configuration. However, the framework would generalize to other approaches of visualization recommendation that may have a different way extracting a visualization instance. Given a single dataset from the corpus that has a set of relevant visualizations, we construct positive (relevant) visualizations as in Figure 1, and complement with “negative” (or non-relevant) visualizations.

While it is intuitive to think that non-relevant visualizations in our problem are visualizations that users do not create, such set of non-relevant instances does not naturally exist in the corpus. The corpus only contains visualizations that users do create. Our framework needs to compute non-relevant visualizations on-the-fly from the dynamic space of visualizations that depends on each dataset. In other words, given a different dataset, there is a different set of non-relevant visualizations since the underlying data in the actual visualizations is different. Our framework follows Def. 5 to achieve that. Moreover, the space of non-relevant visualizations is typically

exponential with a size that easily exceeds several thousands. It is difficult to train with a large number of non-relevant visualizations along with a much smaller set of relevant visualizations. Our framework follows Def. 6 to uniformly sample a fixed number of non-relevant visualizations from the same dataset. Our framework also welcomes other ways of sampling non-relevant visualizations, e.g. drawing non-uniform samples from the pool of non-relevant visualizations (e.g. based on the popularity of the configuration in a visualization, or biased to sample most-similar or least-similar non-relevant visualizations).

#### 5.4 Testing and Deployment

Now, we discuss how a visualization recommendation model  $\mathcal{M}$  learned from the training set of relevant and non-relevant visualizations can be used for testing and deployment. Given a new or selected held-out dataset from the corpus, the model outputs a list of recommended visualizations for the specific dataset. Different from traditional recommender systems where the space of items are shared for all users (including new users), our framework generates a ranking of visualizations to recommend with on-the-fly, which are dependent to the dataset. As the entire space of visualizations for each dataset can be large, negative sampling of non-relevant visualizations allow us to test on more datasets efficiently and derive evaluation metrics without losing statistical rigor. We describe the evaluation metrics for our visualization recommendation problem in Sec. 5.5. When the model gets deployed and tested on a new dataset, the recommended visualizations are selected from the entire space of visualizations for that dataset (as in Def. 3).

#### 5.5 Evaluation Metrics

Evaluating the quality of the ranking of visualizations (recommendations) given by the learned model has its unique challenges. We summarize the challenges and propose a suitable evaluation metric.

In traditional recommender systems, there is a shared global set of items to recommend to any user. However, in visualization recommendation, the set of visualizations to be recommended is generated on-the-fly based on the dataset of interest. Since each dataset gives rise to a new set of visualizations that can be recommended, we therefore must evaluate the quality of ranking for each individual dataset and explicitly account for the different space of visualizations being ranked for every different dataset  $X_i$ . Therefore, the standard ranking metrics such as nDCG cannot be used directly for visualization recommendation. This evaluation must be performed completely independent of any other dataset in the corpus. For instance, in traditional recommender systems, we simply use a model to infer scores for every item since all items are shared by all users. However, in visualization recommender systems, suppose we want to evaluate whether the model can rank actual relevant/positive visualizations from a held-out test dataset highly, then we have to generate all possible visualizations for the specific dataset, and then compute the ranking metric over this set of visualizations independently of other visualizations from other datasets. As such, we have to repeat this process for every dataset, correct for the difference in space, and then average the result.

Furthermore, the number of possible visualizations the ranking is computed over depends entirely on the dataset and the number of

attributes in it. Therefore, the difficulty of the visualization ranking problem varies based on the dataset, and more specifically, the number of attributes in that dataset. For instance, it is easy to score a high nDCG for a dataset with only two attributes as opposed to one with hundreds.

For these reasons, the traditional ranking metrics (e.g., nDCG) are not appropriate for the visualization recommendation problem and give incorrect and misleading results. This leads us to propose a modified version of nDCG that can be used for evaluation of ML-based visualization recommendation models. As an aside, other evaluation metrics can also be corrected in a similar fashion. Given  $N$  test datasets along with  $N$  sets of held-out positive visualizations  $\{\mathbb{V}_i\}_{i=1}^N$ , then we propose a modified nDCG defined formally as:

$$nDCG@K = \frac{1}{N} \sum_{i=1}^N \frac{1}{Z_i^K} \sum_{j=1}^K \frac{2^{Y_{ij}} - 1}{\log_2(j+1)} \quad (36)$$

$$Z_i^K = \sum_{j=1}^{\min(K, |\mathbb{V}_i|)} \frac{1}{\log_2(j+1)} \quad (37)$$

where  $j$  is the rank,  $Y_{ij} \in \{0, 1\}$  is the ground-truth label (relevant/irrelevant) of the visualization at position  $j$  in the ranking of visualizations for dataset  $X_i$ , and  $Z_i^K$  is the normalization factor for dataset  $X_i$ . The dataset-dependent normalization factor  $Z_i^K$  ensures that a perfect ranking for our visualization recommendation problem receives a perfect score of 1. This is required since each dataset  $X_i$  may have a different number of positive visualizations  $|\mathbb{V}_i|$ , that is, for any arbitrary two datasets  $X_i$  and  $X_j$ ,  $|\mathbb{V}_i| \neq |\mathbb{V}_j|$ . The perfect ranking recommends all (but no more than  $K$ ) positive visualizations at the top. Our modified  $nDCG$  emphasizes the quality of the visualization ranking at the top of the list of recommended visualizations for each dataset  $X_i$  since  $1/\log_2(j+2)$  decreases quickly and then asymptotes to a constant as  $j$  increases. Therefore, a good end-to-end learning-based visualization recommender system must be able to give up some of its performance at the bottom of the list of recommended visualizations to improve the performance at the top. In Sec. 6.1, we use the proposed  $nDCG$  metric from Eq. 36 up to the 20th position.

## 6 EXPERIMENTS

In this section, we conduct experiments to answer the following research questions:

- **RQ1:** Given an arbitrary and unseen user selected data set, is our learned model able to automatically recommend the top visualizations that are most important to the user, *i.e.*, the visualizations they manually created, which are held-out for evaluation (Sec. 6.1)?
- **RQ2:** Does our proposed wide-and-deep approach outperform common-sense baselines for end-to-end visualization recommendation? Is the best performance achieved when using the full wide-and-deep model or do the simpler variants of our approach, namely, using the wide-only or deep-only component of our model perform better (Sec. 6.1)?
- **RQ3:** Do human experts prefer our ML-based visualization recommendations or the ones from the rule-based system, Voyager2, that uses CompassQL (Sec. 6.2)?

**Table 4: Training Corpus Statistics.** # CONFIG/DATASET denotes the average number of configurations used by each dataset.

#DATASETS	#Vis. CONFIGS	#ATTRIBUTES	#VISUALIZATIONS	#ATTRIBUTE/DATASET	# Vis. CONFIGS/DATASET
925	60	11,778	4,865	11.93	5.89

**Table 5: Quantitative Results for Visualization Recommendation.** See text for discussion.

Model	nDCG					RANK
	@1	@2	@5	@10	@20	
Random	0.207	0.206	0.253	0.311	0.457	5
ConfigPop	0.366	0.532	0.671	0.691	0.693	4
Ours	<b>0.827</b>	<b>0.827</b>	<b>0.867</b>	<b>0.882</b>	<b>0.897</b>	1
Ours (Deep-only)	0.804	0.807	0.851	0.866	0.887	2
Ours (Wide-only)	0.721	0.714	0.768	0.801	0.839	3

- **RQ4:** Is the learning-based visualization recommendation system able to learn general rules (which would be preferred by human experts) from the large training corpus of datasets and user-generated visualizations (Sec. 6.3)?

To answer RQ1 and RQ2, we quantitatively evaluate our ML-based visualization recommendation system in Section 6.1. For RQ3, we perform a user study in Section 6.2 comparing the effectiveness of our ML-based visualization recommender system to the state-of-the-art rule-based system called CompassQL, which is used in both Voyager and Voyager2. Finally, in Section 6.3, we show a number of examples that demonstrate the ability of our ML-based approach to learn rules from the training corpus, and in many cases, perform better than CompassQL (used in Voyager2), and therefore, overcome many of the limitations that exist in such rule-based systems. These examples also show that our model does not require any manual effort to define such rules, but can automatically learn them from the visualization corpus used for training our model.

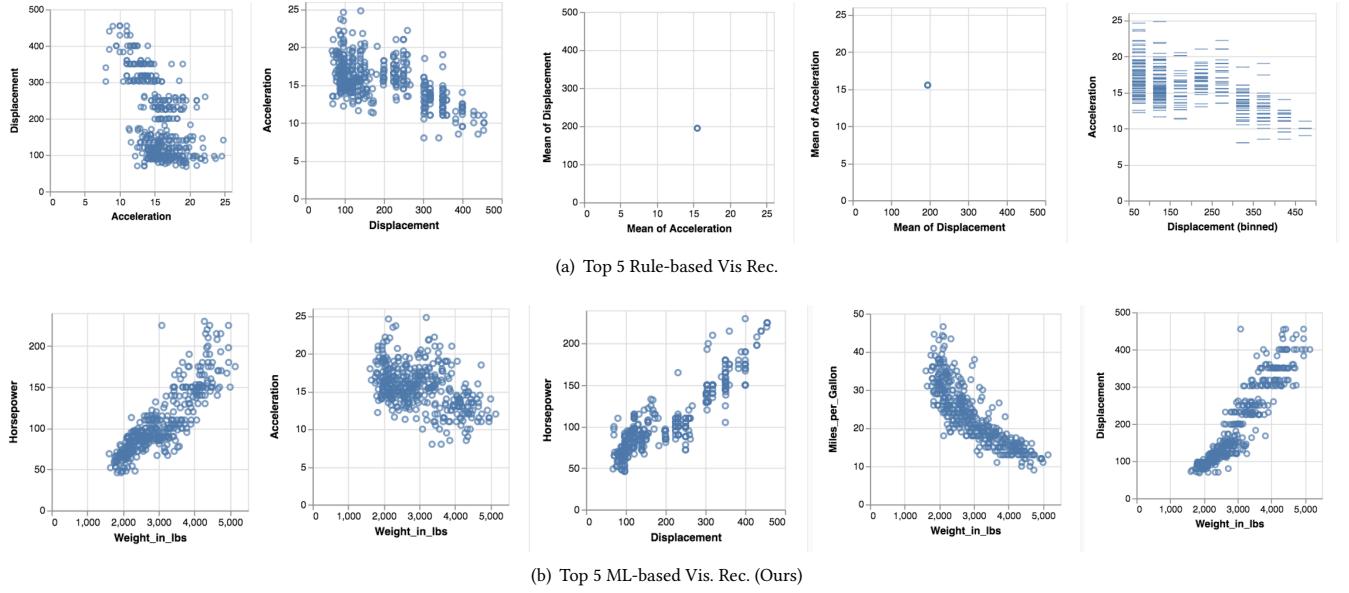
## 6.1 Quantitative Results

In this section, we answer RQ1 and RQ2 by evaluating our ML-based visualization recommendation system quantitatively. For quantitative evaluation, we use the evaluation framework proposed in Section 5. For training our ML-based models, we use a training corpus of 1K datasets (and their visualizations) from the *Plot.ly* corpus [3]. We provide the statistics of the training corpus used for learning our model in Table 4. Notably, as shown in Table 4, there are roughly 1K datasets that have an average of about 12 attributes each. Our ML-based model for visualization recommendation is learned using 1K datasets consisting of about 12K attributes and about 5K user-generated visualizations that use some of the 12K attributes.

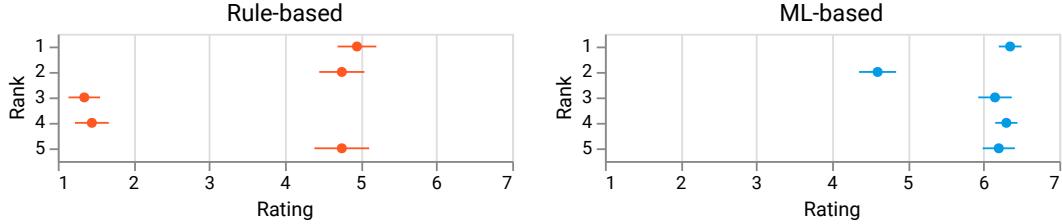
Since this work proposes the first end-to-end learning-based visualization recommender system, we compare our approach using two common-sense baselines along with two simpler variants of our model. The first common-sense baseline is called random, and refers to a method that simply recommends the top-k visualizations chosen uniformly at random from the set of generated visualizations for the specific dataset. This baseline is important for understanding if our wide-and-deep learning approach is able to

learn something meaningful from the raw data (e.g., what visualization preferences, like chart types and so forth users prefer for data with certain characteristics, or what makes a visualization better than another one, and so forth) and if the model is meaningful and useful for visualization recommendation or if it performs no better than random. We also propose another common-sense baseline for evaluating ML-based visualization recommender systems based solely on the popularity (or frequency) of a visualization configuration in the training corpus. We call this baseline ConfigPop. As an aside, the notion of a visualization configuration, which is proposed in this work, is essentially an abstraction of a visualization, consisting of all the design choices, but not the actual data (or attribute names) used in the visualization, see Section 3 for a more formal definition of the proposed notion of a visualization configuration.

To evaluate the effectiveness of the top recommended visualizations, we use the modified normalized Discounted Cumulative Gain (nDCG) at  $k \in \{1, 2, 5, 10, 20\}$  as in Sec. 5.5 for the different top- $k$  visualization recommendations (nDCG@ $k$ ). Results are reported in Table 5. Strikingly, our wide-and-deep learning-based model for visualization recommendation performs the best, achieving a high nDCG across all  $k = 1, \dots, 20$ , as shown in Table 5. Hence, this confirms that our ML-based visualization recommendation model accurately learns to recommend the top visualizations that are most important to the user, despite that the model has never seen the dataset nor the visualizations created by that user before (RQ1). In addition, we observe that our full wide-and-deep learning-based approach and the simpler model variants of our approach always outperform the other methods across all  $k \in \{1, 2, 5, 10, 20\}$ . Furthermore, our approach with both the wide and deep components, has the highest nDCG scores compared to the two common-sense baselines, and our two ablation model variants that use only the wide or deep components of our model, as shown in Table 5 (RQ2). It is also important to note that results at smaller  $k$  are obviously more important, and these are exactly the situations where our models and the variants perform extremely well compared to the others. As an example, at nDCG@1, our wide-and-deep learning visualization recommendation model achieves 0.827 at  $k=1$ , whereas the best baseline is only able to achieve an nDCG of 0.366. This is an improvement in nDCG of 124% over the best baseline at  $k=1$ .



**Figure 3: Comparing the top-5 visualization recommendations from the existing end-to-end rule-based system (Voyager2 using CompassQL) to our end-to-end ML-based visualization recommendation system. See text for discussion.**



**Figure 4: Human experts' ratings on top 5 visualizations from rule-based (Voyager2 using CompassQL) and our ML-based systems. Most visualizations from the ML-based system received higher ratings than rule-base system. Strikingly, the top-1 ranked visualization from human experts exactly matches the top-1 visualization recommended by our ML-based model. Furthermore, the top-4 visualizations receiving the highest rating by human experts are those from the ML-based system. See the text for discussion on other important findings.**

This result clearly demonstrates the effectiveness of our end-to-end ML-based visualization recommendation model as it is able to effectively recover the held-out ground-truth visualization that was generated and therefore preferred by an actual user. Furthermore, the model is also able to distinguish between visualizations that were not preferred by a user, as they receive a lower score from the ML-based recommendation model. These results and findings confirm that our model learns to recommend high quality visualizations that a user will likely prefer from an arbitrary and unseen user-selected data set. From Table 5, we also observe that while the full wide-and-deep learning model outperforms our other model variants, the second best performing model is our deep only variant, and it achieves better performance than the wide-only variant across all  $k$ . Finally, our ML-based visualization recommendation models always outperform the other baselines across all  $k$ , and thus are the top 3 best performing models followed by the other baselines

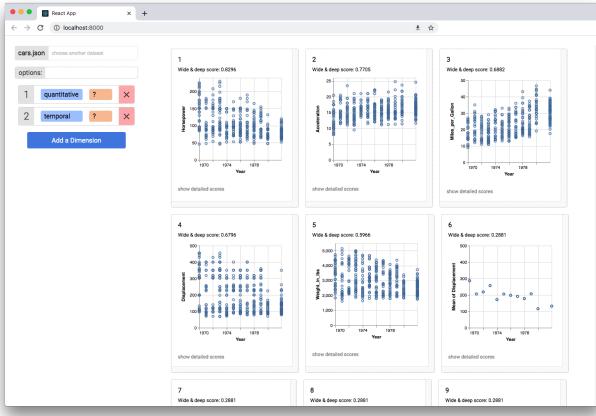
that lack any machine learning (using essentially rules, e.g., config-Pop always predicts the most popular visualization configuration for a given data set).

## 6.2 User Study

In this section, we perform a user study to compare our end-to-end ML-based visualization recommender system to the existing end-to-end system that uses rules as opposed to learning.<sup>13</sup> For this, we take the top 5 recommended visualizations from the rule-based system (Voyager2 using CompassQL) and the top-5 recommended visualizations from our ML-based system (for the standard car data). The top-5 from the rule-based and our ML-based end-to-end visualization recommender system is shown in Figure 3. Given the set of 10 visualizations, we randomize the order by taking a uniform

<sup>13</sup>The end-to-end rule-based system used in this study is Voyager2, which uses CompassQL.

## ML-based Visualization Recommendation: Learning to Recommend Visualizations from Data



**Figure 5: System screenshot of our end-to-end ML-based visualization recommendation approach. It receives a dataset as input, and shows the top recommended visualizations from our approach ranked by the scores.**

random permutation of them, and then display them to the human experts in this order. Human experts are asked to assign a score to each visualization in 7-point Likert scale. Afterwards, we compute the overall score of a visualization by taking the mean of the scores assigned by the experts. In this study, there were 21 human experts rating the top-5 visualizations from either system using a 7-point Likert scale.

Comparing the top-5 recommended visualizations from either system, human experts gave significantly higher scores to those visualizations that our ML-based system recommended. Hence, we observed a strong preference by the human experts towards the visualizations recommended by our system as opposed to the rule-based system. Overall, the human experts assigned a mean score of 5.92 to the top-5 visualizations from the ML-based approach and a mean score of 3.45 to those from the rule-based approach (RQ3). Hence, we can clearly see that the ML-based visualization recommendations are significantly better than the rule-based approach. This result is significant at  $p\text{-val}=0.01$ . We also provide the mean score and the  $\pm$ variance for each of the top-5 visualizations from either system in Figure 4. Strikingly, the top ranked visualization by the human experts is exactly the top ranked visualization from our ML-based system. Furthermore, among the 10 visualizations that human experts scored, the top 4 visualizations with the highest score are those from our ML-based visualization recommendation system, and not from the rule-based system. As shown in Figure 4, the variance of the ML-based recommendations are nearly always less than the rule-based system. This difference is also significant. We posit that this is due to the discrete nature of the rule-based recommender system that scores visualization in a discrete fashion using manually defined rules, and so even though the visualization may appear to be high quality with respect to the manually defined rules, it is not of high quality with respect to the actual data and insights that visualizations seek to show from the data.

### 6.3 Qualitative Analysis

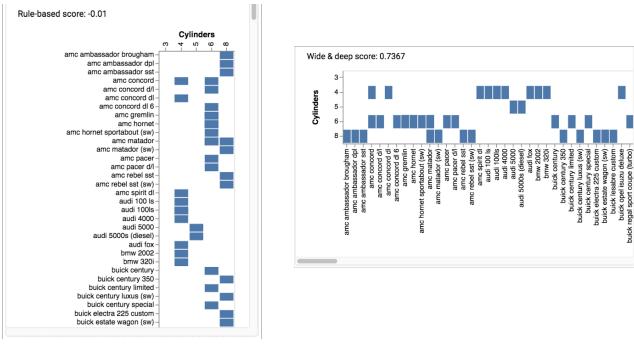
While Section 6.1 demonstrated the effectiveness of the visualization ranking from our ML-based approach using a quantitative ranking evaluation metric whereas Section 6.2 revealed that human experts overwhelmingly preferred visualizations recommended by the ML-based visualization recommendation model compared to those from the state-of-the-art rule-based approach. In this section, we perform a case study to investigate whether the ML-based visualization recommendation model is able to learn meaningful visual rules from the large training corpus.

To investigate the effectiveness of the visualization recommendations given by our end-to-end ML-based approach, we compare with the existing end-to-end rule-based approach (Voyager2). As shown in Figure 5, we developed an interface for our ML-based visualization recommendation system that allows the user to select or upload a dataset of interest, and then we automatically recommend them the top visualizations for that given dataset using the learned model. The recommended visualizations are then displayed to the user in order of relevance/importance score which is inferred from our ML-based model. In addition, the user can specify different queries interactively using the interface. For instance, they can select the attribute types (*quantitative*, *nominal*, and *temporal*) that they want to visualize, and the system immediately infers and displays the top most relevant recommended visualizations to the user. As an aside, the user can also select attributes of interest to include in the recommended visualizations, aggregations to use, chart-types, and so on. This is similar to the interface used by Voyager2.

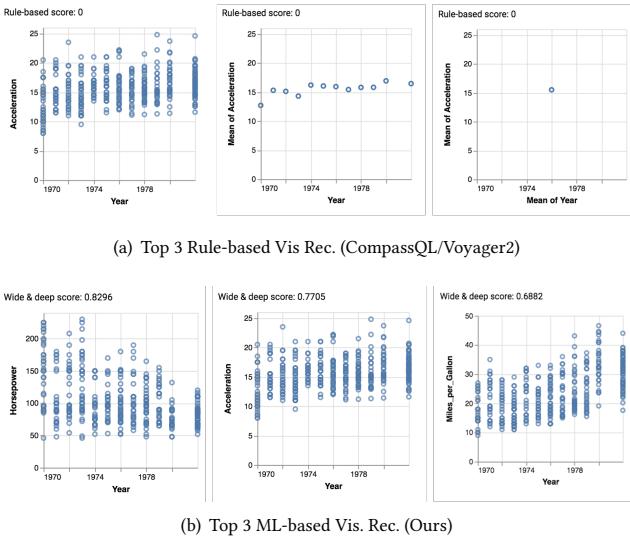
In this case study, we use the cars dataset about car specifications, and specify a few queries. Results reveal several aspects where our end-to-end ML-based visualization recommendation approach is more effective than the end-to-end rule-based approach (Voyager2), and the ability of our system to automatically learn to recommend visualizations that would be preferred by even domain experts, without the manual specification of any rules (RQ4).

**6.3.1 Learning to place attributes like an expert.** Now we demonstrate how our ML-based system is able to learn to prefer visualizations that a human expert would also prefer. Using a query for visualizations containing two nominal attributes, we see that both approaches recommend visualizations with the attribute *car model name* in their top visualizations as shown in Figure 6. However, the rule-based system incorrectly recommends a visualization with a vertical layout whereas the ML-based approach learns to recommend a horizontal layout and penalizes the vertical charts. Interestingly, the ML-based model is able to learn the fact that domain experts prefer to do these types of charts horizontally, as opposed to vertically, and therefore our wide-and-deep learning model penalizes the vertical charts to ensure they are not recommended to the user. This is in complete contrast to the rule-based approach, which seems to favor vertical layout, despite that human experts would recommend against such charts.

**6.3.2 Tie-breaking issues.** A key fundamental problem with the existing rule-based visualization recommender system (Voyager2 using CompassQL) is that visualizations are often scored using a set



**Figure 6: Top visualizations from rule-based (left) vs. our ML-based approach (right) for a query on two nominal attributes. The ML-based approach learns to recommend a horizontal layout and penalizes the vertical charts. This is consistent with the fact that domain experts prefer to do it horizontally.**



**Figure 7: Tie-breaking issues of the rule-based approach where all visualizations are scored the same and thus the system fails to find an appropriate high quality ranking. In comparison, our wide-and-deep learning approach is able to learn more meaningful scores that appropriately differentiate between the most important and insightful visualizations and those that are less important.**

of manually defined rules that assign simple discrete scores to visualizations that either satisfy or violate the manual rules defined by people. This oversimplified scoring results in many visualizations having exactly the same score, and therefore, no way to actually rank them. In this case, the existing end-to-end system simply displays the visualizations in the order they were generated, which is most often not very appropriate. In Figure 7, we show one such case of this where the top visualizations from the rule-based are all assigned the same score of 0, and thus fails to prefer one over

the other and vice-versa. Moreover, while we show only three visualizations here with score 0, there are even more further down the list that also have score of 0. Hence, in this case, the rule-based system simply breaks ties randomly, and often ends up with low quality visualizations that are ranked higher than some visualizations that are clearly better. In comparison, our wide-and-deep learning approach is able to learn more meaningful scores that appropriately differentiate between the most important and insightful visualizations and those that are less important. For instance, our ML-based visualization recommendation model assigns a score to the visualization shown in the 3rd position by the rule-based approach, which is clearly not a useful visualization, especially compared to the exponential amount of other possibilities. There are many other similar situations where our scores are significantly more useful and completely avoid the tie-breaking issues of the rule-based approach.

## 7 CONCLUSION

In this work, we proposed the first end-to-end ML-based visualization recommendation system. We first formalized the ML-based visualization recommendation problem and described a generic learning framework for solving it. Next, we proposed a wide-and-deep learning architecture that combines a wide component with a deep learning component. Each visualization is scored with the input of two parts, an attribute combination and a visualization configuration, using our wide-and-deep learning visualization recommendation model. Given a new unseen dataset from an arbitrary user, the learned model is used to automatically generate, score, and output a list of recommended visualizations for that specific dataset. Further, we present an evaluation framework that can evaluate visualization recommendation system learned from a large corpus of visualizations and datasets. We demonstrated the effectiveness of the proposed approach in three different ways. First, we used the evaluation framework to quantitatively demonstrate the effectiveness of the ranking of visualizations from our ML-based visualization recommendation system. Second, we also validated the effectiveness of our ML-based system through a user study of 20 human experts, and found that the top-1 ranked visualization from human experts matched exactly the top-1 visualization from our system. Most importantly, human experts overwhelming preferred the ML-based visualization recommendations over the existing rule-based system as shown in Section 6.2. Third, we also performed qualitative analysis on the recommendations from our ML-based model and the state-of-the-art rule-based approach, and discussed many different advantages where our model is able to learn visual rules from the large training corpus that are not even incorporated in the rule-based approach.

## REFERENCES

- [1] Humaira Ehsan, Mohamed A Sharaf, and Panos K Chrysanthis. 2016. Muve: Efficient multi-objective view recommendation for visual data exploration. In *ICDE '16*. 731–742.
- [2] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR '17*. 355–364.
- [3] Kevin Hu, Michiel A Bakker, Stephen Li, Tim Kraska, and César Hidalgo. 2019. Vizml: A machine learning approach to visualization recommendation. In *CHI '19*. 1–12.
- [4] Kevin Hu, Diana Orghian, and César Hidalgo. 2018. Dive: A mixed-initiative system supporting integrated data exploration workflows. In *Proceedings of the*

**ML-based Visualization Recommendation:**  
Learning to Recommend Visualizations from Data

- Workshop on Human-In-the-Loop Data Analytics.* 1–7.
- [5] Doris Jung-Lin Lee. 2020. *Insight Machines: The Past, Present, and Future of Visualization Recommendation*.
  - [6] Doris Jung-Lin Lee, Himmel Dev, Huizi Hu, Hazem Elmeleegy, and Aditya Parameswaran. 2019. Avoiding drill-down fallacies with VisPilot: assisted exploration of data subsets. In *IUI '19*. 186–196.
  - [7] Jock Mackinlay. 1986. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.* 5, 2 (1986), 110–141.
  - [8] Jock Mackinlay, Pat Hanrahan, and Chris Stoltz. 2007. Show me: Automatic presentation for visual analysis. *IEEE transactions on visualization and computer graphics* 13, 6 (2007), 1137–1144.
  - [9] Dominik Moritz, Chenglong Wang, Greg L Nelson, Halden Lin, Adam M Smith, Bill Howe, and Jeffrey Heer. 2018. Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2018), 438–448.
  - [10] Daniel B Perry, Bill Howe, Alicia MF Key, and Cecilia Aragon. 2013. VizDeck: Streamlining exploratory visual analytics of scientific data. In *iConference '13*.
  - [11] Steven F Roth, John Kolojejchick, Joe Mattis, and Jade Goldstein. 1994. Interactive graphic design using automatic presentation knowledge. In *CHI '94*. 112–117.
  - [12] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. 1998. Discovery-driven exploration of OLAP data cubes. In *International Conference on Extending Database Technology*. Springer, 168–182.
  - [13] Manasi Vartak, Silu Huang, Tarique Siddiqui, Samuel Madden, and Aditya Parameswaran. 2017. Towards visualization recommendation systems. *ACM SIGMOD Record* 45, 4 (2017), 34–39.
  - [14] Leland Wilkinson, Anushka Anand, and Robert Grossman. 2005. Graph-theoretic scagnostics. In *IEEE Symposium on Information Visualization*. IEEE, 157–164.
  - [15] Graham Wills and Leland Wilkinson. 2010. Autovis: automatic visualization. *Information Visualization* 9, 1 (2010), 47–69.
  - [16] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2015. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE transactions on visualization and computer graphics* 22, 1 (2015), 649–658.
  - [17] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Towards a general-purpose query language for visualization recommendation. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. 1–6.
  - [18] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2017. Voyager 2: Augmenting visual analysis with partial view specifications. In *CHI '17*. 2648–2659.