Learning to Recommend Visualizations from Data

Xin Qian
University of Maryland

Ryan A. Rossi
Adobe Research

Fan Du
Adobe Research

Sungchul Kim
Adobe Research

Eunyee Koh
Adobe Research

Sana Malik
Adobe Research

Tak Yeon Lee
KAIST

Joel Chan
University of Maryland

ABSTRACT

Visualization recommendation is important for exploratory analysis and making sense of the data quickly by automatically recommending relevant visualizations to the user. In this work, we propose the first end-to-end ML-based visualization recommendation system that leverages a large corpus of datasets and their relevant visualizations to learn a visualization recommendation model automatically. Then, given a new unseen dataset from an arbitrary user, the model automatically generates visualizations for that new dataset, derives scores for the visualizations, and outputs 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.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Human-centered computing** → **Visualization systems and tools**.

KEYWORDS

Visualization recommendation; deep learning; data visualization

ACM Reference Format:
Xin Qian, Ryan A. Rossi, Fan Du, Sungchul Kim, Eunyee Koh, Sana Malik, Tak Yeon Lee, and Joel Chan. 2021. Learning to Recommend Visualizations from Data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467224

1 INTRODUCTION

Visualization recommendation is the problem of automatically generating, scoring, and recommending the most relevant visualizations for a dataset of interest to the user, and can improve productivity by reducing the time required by analysts to first find interesting insights and then visualize them. Previous end-to-end visualization recommendation systems are rule-based, leveraging a small set of rules hand-crafted by domain experts to score the generated visualizations [41, 43]. As such, these rule-based systems have many issues that our proposed ML-based visualization recommendation approach addresses. First, these systems often have quality issues, limiting the utility and usefulness of the recommended visualizations. Second, many of the visualizations receive the same exact score, and therefore, these rule-based systems are unable to appropriately differentiate between the visualizations receiving the same heuristic score. Finally, they are not data-driven nor automatic, and therefore unable to adapt automatically based on changing visualization and data preferences of the user. Instead, adding rules is a tedious and costly manual process.

Some recent work has explored the use of machine learning for subtasks of visualization recommendation. For instance, VizML [14] used machine learning to predict the type of a chart (*e.g.*, bar, scatter) instead of a complete visualization, and Draco [26] used a model to infer weights for a set of manually defined rules. However, there are no systems yet that use machine learning for end-to-end visualization recommendation of complete visualizations for any arbitrary new unseen dataset of interest to a user.

In this work, we propose the first *end-to-end ML-based visualization recommendation system* that automatically learns a model $\mathcal{M}$ from a large training corpus $\{\mathbf{X}_i, \mathbb{V}_i\}_{i=1}^{N}$ of N training datasets and the corresponding N sets of user-generated visualizations. Given a new user-selected dataset of interest, the learned model $\mathcal{M}$ can automatically score and recommend the top-k most relevant visualizations for that 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 provide us with everything required to characterize a visualization. Next, we propose a wide-and-deep learning model [4, 12] for visualization recommendation based on this problem formulation that learns from the attribute selections and their visualization configurations. The wide component learns from sparse attribute meta-features and sparse visualization configuration features, whereas the deep component learns from dense representations of the meta-features of the attributes

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467224

and visualization configurations. Scores from both components are then combined to obtain the final score of a complete visualization. We also describe a framework for quantitatively evaluating learning-based systems and provide a sound and appropriate rank-based visualization evaluation metric for quantitatively comparing such systems. The evaluation framework will be useful for making further progress in developing better and more accurate end-to-end visualization recommendation systems that leverage machine learning. Finally, we demonstrate the effectiveness and utility of the proposed end-to-end ML-based visualization recommendation system through a set of comprehensive experiments.

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

- **ML-based Problem Formulation**: We formally introduce the end-to-end ML-based visualization recommendation problem, and present a generic learning framework for it.
- **Wide & Deep ML-based Visualization Recommender**: We propose a wide-and-deep approach that learns a model $\mathcal{M}$ from a training corpus of datasets and relevant visualizations. The model can then be used to automatically recommend the most relevant visualizations for arbitrary unseen datasets of interest.
- **Effectiveness**: We demonstrate the effectiveness of our approach through a comprehensive set of experiments including quantitative evaluation of the visualization recommendations (Sec. 5.1), a user study comparing our ML-based system to the state-of-the-art rule-based system (Sec. 5.2), and a case study (Sec. 5.3).

2 RELATED WORK

Rule-based systems such as Voyager [37, 41, 43], VizDeck [28], Draco [21, 26] and DIVE [15] incorporate visual encoding rules to assist user data exploration [7, 18, 24, 25, 31, 36]. 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 [24], 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 datasets 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 for each 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. As user study has been a prevailing evaluation approach in visualization [13, 15, 17, 23, 27, 32, 34, 35], many systems require experimenting with human users to validate the manually defined rules. In comparison, our work learns a visualization recommendation model $\mathcal{M}$ directly from a large corpus of training data, in a fully automatic data-driven fashion. The trained model $\mathcal{M}$ can then be used to recommend entire visualizations from any arbitrary unseen dataset of interest. We also propose an evaluation framework (Appendix Sec. A) to validate the effectiveness of ML-based visualization recommendation models.

A body of related work proposed techniques for sub-tasks in interactive visualization recommendation systems, such as improving expressiveness or perceptual effectiveness [24], matching user intents [10], displaying trends and outliers [20], etc. These sub-tasks can generally be divided into two categories [18, 42]: whether the solution focuses on recommending data (*what data to visualize*), such as Discovery-driven Data Cubes [33], Scagnostics [39], AutoVis [40], AutoPartition [2], Foresight [6], SpotLight [11], ScagExplorer [5], VisPilot [19], VizDeck [28] and MuVE [9] or recommending encoding (*how to design and visually encode the data*), such as APT [24], ShowMe [25], BDVR [10], SeeDB [38], Draco–learn [26], and LQ^2 [44]. While some of those are ML-based [14, 19, 26, 28], none recommends entire visualizations, and thus does not solve the visualization recommendation problem that lies at the heart of our work. VizML [14] used machine learning to predict the type of a chart (*e.g.*, bar, scatter, etc.) instead of complete visualization. Another work Draco [26] used a model to infer weights for a set of manually defined rules. VisPilot [19] recommended different drill-down data subsets from datasets. Data2Vis [8] learned a seq2seq model based on Vega-specs for a few datasets, and captured mostly the design choices (as opposed to the data attributes as well, as done in our work). 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 system that is completely automatic and data-driven. A preprint of this work first appeared in 2020 [30].

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 $\{\mathbb{V}_i\}_{i=1}^{N}$,[1] 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 ineffective visualizations.[2]
- **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.[3]

The visualization recommendation training data $\mathcal{D} = \{\mathbf{X}_i, \mathbb{V}_i\}_{i=1}^{N}$ can be general and collected from a variety of different sources. 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 an effective visualization learned by

[1] $\mathbb{V}_i$ is the set of visualizations associated with the ith dataset $\mathbf{X}_i$.

[2] 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.

[3] Note each visualization uses a subset of the attributes from the dataset, and some attributes may never be used.

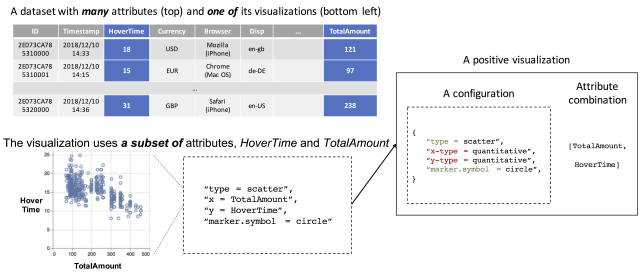


Figure 1: Extracting positive visualizations for training. The left figure shows a dataset from the corpus. It has a set of visualizations in the corpus where each uses a small subset of attributes from the dataset. The right figure shows that the extracted positive visualization decomposes into a vis. configuration and subset of attributes used.

the model will also be 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. Every visualization uses a set of attributes $\mathbf{X}_i^{(k)}$ from a dataset $\mathbf{X}_i$. We now define the space of attribute sets $\boldsymbol{\mathcal{X}}_i = \{\mathbf{X}_i^1, \ldots, \mathbf{X}_i^{(k)}, \ldots\}$ for an arbitrary dataset $\mathbf{X}_i$.

Definition 1 (Space of Attribute Combinations). *Given an arbitrary dataset matrix $\mathbf{X}_i$, let $\boldsymbol{\mathcal{X}}_i$ denote the space of attribute combinations of $\mathbf{X}_i$ defined as*

$$\Sigma : \mathbf{X}_i \rightarrow \boldsymbol{\mathcal{X}}_i, \quad s.t. \tag{1}$$

$$\boldsymbol{\mathcal{X}}_i = \{\mathbf{X}_i^{(1)}, \ldots, \mathbf{X}_i^{(k)}, \ldots\}, \tag{2}$$

where Σ is an attribute combination generation function and every $\mathbf{X}_i^{(k)} \in \boldsymbol{\mathcal{X}}_i$ is a different subset of attributes from $\mathbf{X}_i$.

Let $|\mathbf{X}_i|$ and $|\mathbf{X}_j|$ denote the number of attributes (columns) of two arbitrary datasets $|\mathbf{X}_i|$ and $|\mathbf{X}_j|$, respectively.

Property 1. *If $|\mathbf{X}_i| > |\mathbf{X}_j|$, then $|\boldsymbol{\mathcal{X}}_i| > |\boldsymbol{\mathcal{X}}_j|$.*

The above will be important later when characterizing the space of possible visualizations from a given dataset.

Definition 2 (Space of Visualization Configurations). *Let $\boldsymbol{\mathcal{C}}$ denote the space of all visualization configurations such that a visualization configuration $C_{ik} \in \boldsymbol{\mathcal{C}}$ 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 $\mathbf{X}_i$, we replace it with its type. Therefore visualization configurations are essentially visualizations without any attribute (data).*

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

The above implies that $C_{ik} \in \boldsymbol{\mathcal{C}}$ 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 $\mathbf{X}_i$, then it is replaced by its type (*i.e.*, numerical, categorical, ordinal, etc.).

A visualization configuration *and* the attributes selected is everything necessary to generate a visualization. See Figure 1 for an example. The space of visualization configurations is large since it arises from all possible combinations of design choices and their values, *e.g.*, mark/chart (bar, scatter, ...), x/y-type (numerical, categorical, ordinal, temporal, ..., none), x/y-aggregate (sum, mean, bin, ..., none), size (1pt, 2pt, ...), color, and so on.

Definition 3 (Space of Visualizations of $\mathbf{X}_i$). *Given an arbitrary dataset matrix $\mathbf{X}_i$, we define $\mathbb{V}_i^\star$ as the space of all possible visualizations that can be generated from $\mathbf{X}_i$. More formally, the space of visualizations $\mathbb{V}_i^\star$ is defined with respect to a dataset $\mathbf{X}_i$ and the space of visualization configurations $\boldsymbol{\mathcal{C}}$,*

$$\boldsymbol{\mathcal{X}}_i = \Sigma(\mathbf{X}_i) = \{\mathbf{X}_i^1, \ldots, \mathbf{X}_i^{(k)}, \ldots\} \tag{3}$$

$$\xi : \boldsymbol{\mathcal{X}}_i \times \boldsymbol{\mathcal{C}} \rightarrow \mathbb{V}_i^\star \tag{4}$$

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

Hence, given a subset of attributes $\mathbf{X}_i^{(k)} \in \boldsymbol{\mathcal{X}}_i$ from dataset $\mathbf{X}_i$ and a visualization configuration $C \in \boldsymbol{\mathcal{C}}$, then $\xi(\mathbf{X}_i^{(k)}, C)$ is the corresponding visualization. Define $\mathbf{X} \neq \mathbf{Y} \implies \forall i, j\ \mathbf{x}_i \neq \mathbf{y}_j$.

Lemma 1. *$\forall \mathbf{X}_i, \mathbf{X}_j$ s.t. $\mathbf{X}_i \neq \mathbf{X}_j$, then $\xi(\Sigma(\mathbf{X}_i), \boldsymbol{\mathcal{C}}) \cap \xi(\Sigma(\mathbf{X}_j), \boldsymbol{\mathcal{C}}) = \emptyset$.*

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

Definition 4 (Positive Visualizations of $\mathbf{X}_i$). *Given an arbitrary dataset matrix $\mathbf{X}_i$, we define $\mathbb{V}_i$ as the set of positive visualizations (user-generated, observed) from dataset $\mathbf{X}_i$.*

Definition 5 (Negative Visualizations of $\mathbf{X}_i$). *Let $\mathbb{V}_i^\star$ denote the space of all visualizations that arise from dataset $\mathbf{X}_i$ such that the user-generated (positive) visualizations $\mathbb{V}_i$ satisfies $\mathbb{V}_i \subseteq \mathbb{V}_i^\star$, then the* space of negative visualizations for dataset $\mathbf{X}_i$ *is $\mathbb{V}_i^- = \mathbb{V}_i^\star \setminus \mathbb{V}_i$.*

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

Definition 6 (Sampling Negative Visualizations of $\mathbf{X}_i$). *Given dataset $\mathbf{X}_i$, we sample negative (non-relevant) visualizations from $\mathbb{V}_i^- = \mathbb{V}_i^\star \setminus \mathbb{V}_i$ (Def. 5) as follows:*

$$k \sim \mathit{UniformDiscrete}\{1, 2, \ldots, |\mathbb{V}_i^-|\}, \quad \text{for } j = 1, 2, \ldots \tag{5}$$

$$\widehat{\mathbb{V}}_i^- = \widehat{\mathbb{V}}_i^- \cup \mathcal{V}_{ik}^- \tag{6}$$

where $\widehat{\mathbb{V}}_i^- \subseteq \mathbb{V}_i^-$. Hence, $\mathcal{V}_{ik}^-$ denotes the jth negative visualization for dataset $\mathbf{X}_i$ sampled from $\mathbb{V}_i^-$.

The negative visualization space is large and therefore sampling this vast space is required to ensure efficient training and inference.

3.1 Learning Vis. Rec. Model

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

Definition 7 (Learning Vis. Rec. Model). *Given the set of training datasets and relevant visualizations $\mathcal{D} = \{\mathbf{X}_i, \mathbb{V}_i\}_{i=1}^{N}$, the goal is to learn a visualization recommendation model $\mathcal{M}$ by optimizing the following general objective function,*

$$\underset{\mathcal{M}}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{(\mathbf{X}_i^{(k)}, C_{ik}) \in \mathbb{V}_i^- \cup \mathbb{V}_i} \mathcal{L}\Big(Y_{ik} \,\big|\, \Psi(\mathbf{X}_i^{(k)}), f(C_{ik}), \mathcal{M}\Big) \quad (7)$$

where $\mathcal{L}$ is the loss function, $Y_{ik} = \{0, 1\}$ is the ground-truth label of the kth visualization $\mathcal{V}_{ik} = (\mathbf{X}_i^{(k)}, C_{ik}) \in \mathbb{V}_i^- \cup \mathbb{V}_i$ for dataset $\mathbf{X}_i$. Further, $\mathbf{X}_i^{(k)} \subseteq \mathbf{X}_i$ is the combination of attributes used in the visualization. In Eq. 7, Ψ and f are general functions over the attribute combination $\mathbf{X}_i^{(k)} \subseteq \mathbf{X}_i$ and the visualization configuration C_{ik} of the visualization $\mathcal{V}_{ik} = (\mathbf{X}_i^{(k)}, C_{ik}) \in \mathbb{V}_i^- \cup \mathbb{V}_i$, respectively.[4]

For computational tractability, we replace $\mathbb{V}_i^-$ in Eq. 7 with the set $\widehat{\mathbb{V}}_i^-$ of sampled negative visualizations for the ith dataset $\mathbf{X}_i$. The generic learning framework for visualization recommendation shown in Eq. 7 can naturally be used to recover many different types of visualization recommendation models.

Definition 8 (Meta-Feature Function). *Let Ψ 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 important characteristics of $\mathbf{x}$. More formally, $\Psi : \mathbf{x} \rightarrow \mathbb{R}^K$ where $\mathbf{x}$ can be of an arbitrary attribute-type (e.g., numerical, categorical, etc.) and size.*

3.2 Recommending Visualizations via Model

Once we have learned the visualization recommendation model $\mathcal{M}$ (Eq. 7) using the training visualization corpus $\mathcal{D}$, we can use $\mathcal{M}$ to score and recommend the top most important and insightful visualizations generated from any new dataset $\mathbf{X}_{\text{test}} \notin \mathcal{D}$.

Definition 9 (ML-based Visualization Recommendation). *Given $\mathcal{M}$ along with a new dataset $\mathbf{X}_{\text{test}} \notin \{\mathbf{X}_i\}_{i=1}^{N}$ of interest, then*

$$\mathcal{M} : \boldsymbol{\mathcal{X}}_{\text{test}} \times \boldsymbol{\mathcal{C}} \rightarrow \mathbb{R} \quad (8)$$

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

$$\rho\big(\{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_Q\}\big) = \underset{\mathcal{V}_t \in \mathbb{V}_{\text{test}}}{\arg\operatorname{sort}} \; \mathcal{M}(\mathcal{V}_t) \quad (9)$$

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. 7), then $\mathcal{M}(\xi(\Sigma(\mathbf{X}_{\text{test}}), \boldsymbol{\mathcal{C}}))$ where $\boldsymbol{\mathcal{C}}$ is the space of relevant visualization configurations. Notice that $\mathcal{M}(\mathbb{V}_{\text{test}}) = \mathcal{M}(\xi(\Sigma(\mathbf{X}_{\text{test}}), \boldsymbol{\mathcal{C}}))$. Furthermore, 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 $\mathbb{V}_i^\star$ and $\mathbb{V}_j^\star$ denote the space of all possible visualizations that arise from $\mathbf{X}_i$ and $\mathbf{X}_j$ held-out datasets, then $\forall r, s, \mathcal{V}_r \in \mathbb{V}_i^\star \neq \mathcal{V}_s \in \mathbb{V}_j^\star$ holds, since a visualization consists of a subset of attributes (data) and design choices. This 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.

[4] Note Ψ and f can also be learned along with the model $\mathcal{M}$ or learned/defined prior to learning the model $\mathcal{M}$.

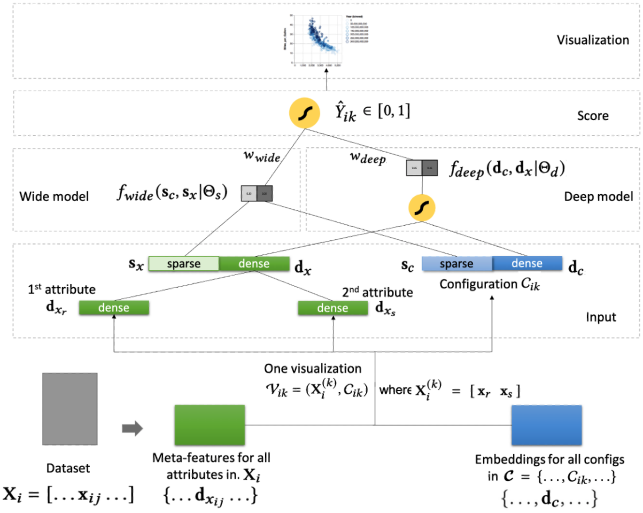


Figure 2: For an arbitrary dataset $\mathbf{X}_i$ (either a new unseen dataset or one from the training corpus), we generate the space of visualizations $\mathbb{V}_i^\star$ for $\mathbf{X}_i$. The visualizations then feed our wide-and-deep network model $\mathcal{M}$ one-by-one. For each visualization, 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.

4 WIDE & DEEP VIS. RECOMMENDATION

Following the general ML-based visualization recommendation formulation proposed in Sec. 3, we now describe our wide-and-deep approach for solving it.

4.1 Wide-and-Deep Approach Overview

We first give a brief overview of the approach.

- **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 (Sec. 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 $f_{wide}(\mathbf{s}_c, \mathbf{s}_x|\Theta_s)$. The *wide* model uses a linear model over 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 $f_{deep}(\mathbf{d}_c, \mathbf{d}_x|\Theta_d)$. The *deep* model uses dense features and non-linear transformations to generalize to unseen feature pairs not 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 gradient-based optimization 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 $\mathbf{X}_i$ [5]. The score $\hat{Y}_{ik}$ is given by

$$\hat{Y}_{ik} = \mathcal{M}(\mathcal{V}_{ik}) = f(\mathbf{X}_i^{(k)}, C_{ik}|\Theta) \in [0, 1] \tag{10}$$

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 Encoding 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 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 Ψ from Def. 8. Since Ψ represents an attribute $\mathbf{x}$ in a shared K-dimensional space, we apply Ψ $\forall \mathbf{x} \in \mathbf{X}_i$.

We propose the meta-feature learning framework, as an instance of Ψ for the network. The framework has several components including a set of nested meta-feature functions Ψ, attribute representation functions, and partitioning functions Π. The framework is summarized in Table 1. For the meta-feature functions used in this work, see Table 3. We first compute the meta-feature functions over different representations of the attribute, $\psi(\mathbf{x}), \psi(p(\mathbf{x})), \psi(g(\mathbf{x})), \ldots$ where $\psi(\mathbf{x})$ are 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 Π that divides a vector $\mathbf{x}$ (or $p(\mathbf{x})$, $g(\mathbf{x})$) of values into k partitions, we can derive meta-features for each partition as follows: $\psi(\Pi_1(\mathbf{x})), \ldots, \psi(\Pi_k(\mathbf{x}))$, $\psi(\Pi_1(p(\mathbf{x}))), \ldots, \psi(\Pi_k(p(\mathbf{x})))$, $\psi(\Pi_1(g(\mathbf{x}))), \ldots, \psi(\Pi_k(g(\mathbf{x})))$ where Π_k denotes the kth partition of values from the partitioning function Π. In this work, we leverage multiple partitioning functions. All the meta-features are then concatenated into a single vector of meta-features describing the characteristics of the attribute $\mathbf{x}$ [29]. More formally, the meta-feature function $\Psi : \mathbf{x} \rightarrow \mathbb{R}^K$ is defined as

$$\begin{aligned}\Psi(\mathbf{x}) = \big[&\psi(\mathbf{x}), \psi(p(\mathbf{x})), \psi(g(\mathbf{x})), \ldots, \psi(\Pi_1(\mathbf{x})), \ldots, \psi(\Pi_k(\mathbf{x})), \ldots, \\ &\psi(\Pi_1(p(\mathbf{x}))), \ldots, \psi(\Pi_k(p(\mathbf{x}))), \ldots, \psi(\Pi_1(g(\mathbf{x}))), \ldots, \psi(\Pi_k(g(\mathbf{x})))\big]\end{aligned} \tag{11}$$

The resulting $\mathbf{d}_x = \Psi(\mathbf{x})$ is a dense *meta-feature* vector of the attribute $\mathbf{x}$. Without loss generality, we also normalize each meta-feature in $\mathbf{d}_x$ via min-max scaling. Our approach is agnostic to the precise meta-feature functions used, and is flexible for use with any alternative set of meta-feature functions (Table 3 in the Appendix).

[5] $\mathbf{X}_i$ 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: Meta-feature framework for an attribute x.

Framework Components	Examples
1. Attribute representations	$\mathbf{x}$, $p(\mathbf{x})$, $g(\mathbf{x})$, $\ell_b(\mathbf{x})$, ...
2. Partitioning values Π	Clustering, binning, quartiles, ...
3. Meta-feature functions ψ	Statistical, info theoretic, ...

We obtain $\Psi(\mathbf{x})$ as the meta-features for each attribute $\mathbf{x}$. An attribute combination $\mathbf{X}_i^{(k)}$ 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 the meta-features $\mathbf{d}_{x_{ij}}$ from each attribute $\mathbf{x}_{ij} \in \mathbf{X}_i^{(k)}$ to get the overall dense feature $\mathbf{d}_x$, written as

$$\mathbf{d}_x = \phi_1(\ldots \mathbf{d}_{x_{ij}} \ldots) = \left[\cdots \mathbf{d}_{x_{ij}} \cdots\right]^T \tag{12}$$

See Figure 2 for an example. It has two attributes selected, *i.e.*, $\mathbf{X}_i^{(k)} = [\, \mathbf{x}_r \;\; \mathbf{x}_s \,]$, 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 $\mathcal{C} = \{\ldots, C_{ik}, \ldots\}$ is shared among 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 $\mathcal{V}_{ik} = (\mathbf{X}_i^{(k)}, C_{ik})$, 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 them to encode C_{ik}.

Definition 10 (Configuration Embedding Function). *Let $\mathcal{H}$ denote a configuration embedding function that maps a configuration C_{ik} to a shared K-dimensional embedding space such that the embedding $\mathcal{H}(C)$ captures the important characteristics of C_{ik} and can be learned along with the model $\mathcal{M}$. More formally, $\mathcal{H} : C_{ik} \rightarrow \mathbb{R}^K$. 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 $\mathcal{H}(\mathcal{C})$ where $\mathcal{H} : \mathcal{C} \rightarrow \mathbb{R}^{K \times M}$.*

For convenience, we denote $\mathbf{d}_c = \mathcal{H}(C_{ik})$ as the dense feature $\mathbf{d}_c$ of the configuration C_{ik}. 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 the configuration embedding function $\mathcal{H}$. This gives, $\mathbf{d}_c = \mathcal{H}(one\_hot(C_{ik}))$ where $\mathcal{H}$ is learnable with the model $\mathcal{M}$.

4.2.3 Complementing dense features with sparse features. Thus far, both the configuration embedding vector $\mathbf{d}_c$ and attribute meta-features $\mathbf{d}_x$ are dense feature vectors. On the other hand, we want our approach to capture feature patterns about the attribute combination $\mathbf{X}_i^{(k)}$ and the configuration C_{ik} that commonly lead to effective visualizations. The 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 ways to derive sparse feature vectors $\mathbf{s}_x$ and $\mathbf{s}_c$. In this work, we compute the sparse feature vector $\mathbf{s}_x$ for attribute combination $\mathbf{X}_i^{(k)}$ by binning the dense meta-feature vector $\mathbf{d}_x$ using a fixed number of n-bins within the range of $[0, 1]$ where each bin has an equal width of $\frac{1}{n}$. Another option is to cluster the dense meta-feature matrix and then one-hot encode the cluster identity of $\mathbf{d}_x$. Our wide-and-deep network is agnostic to the actual option and the precise meta-features that are being used in $\mathbf{d}_x$. In this work, we used the first option. To get the sparse feature $\mathbf{s}_c$ from a configuration embedding vector $\mathbf{d}_c$, we 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 can 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. First, we concatenate $\mathbf{s}_c$ and $\mathbf{s}_x$ into a single sparse vector $\mathbf{s}$ where ϕ_1 is a concatenation operator.

$$\mathbf{s} = \phi_1(\mathbf{s}_c, \mathbf{s}_x) = \begin{bmatrix} \mathbf{s}_c \\ \mathbf{s}_x \end{bmatrix} \tag{13}$$

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

$$\mathbf{s}' = \{\ldots, \phi_k(\mathbf{s}), \ldots\} \tag{14}$$

where $\phi_k(\mathbf{s})$ is the k-th cross-product transformation function,

$$\phi_k(\mathbf{s}) = \prod_{i=1}^{|\mathbf{s}|} \mathbf{s}_i^{t_{ki}}, \quad t_{ki} \in \{0, 1\} \tag{15}$$

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 #3. The cross-product feature $\phi_k(\mathbf{s})$ is 1 if and only if $\mathbf{s}$ 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 $\mathbf{s}'$ are concatenated into $[\mathbf{s}, \mathbf{s}']$, then go through a linear transformation, to get the wide score $f_{wide}(\mathbf{s}_c, \mathbf{s}_x|\Theta_s) \in \mathbb{R}$. More formally,

$$f_{wide}(\mathbf{s}_c, \mathbf{s}_x|\Theta_s) = \mathbf{W}_s^T [\mathbf{s}, \mathbf{s}'] + \mathbf{b}_s \tag{16}$$

where $\mathbf{W}_s^T$ and $\mathbf{b}$ denote the weight matrix and bias vector for the wide model. Further, $\Theta_s = \{\mathbf{W}_s^T, \mathbf{b}\}$ denotes the entire set of parameters in the wide model.

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 corpus $\mathcal{D} = \{\mathbf{X}_i, \mathbb{V}_i\}_{i=1}^N$ yet may lead to effective visualizations. The deep model works by first concatenating the two dense features $\mathbf{d}_c$ and $\mathbf{d}_x$ into an intermediate vector $\mathbf{d} = \phi_1(\mathbf{d}_c, \mathbf{d}_x)$, such that it incorporates the information from both the configuration and the attribute combination. Next, $\mathbf{d}$ is fed into a total of L hidden layers (standard MLP layers). 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. The activation function a_k 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 $f_{deep}(\mathbf{d}_c, \mathbf{d}_x|\Theta_d) \in \mathbb{R}$. More formally,

$$\mathbf{d}_0 = \mathbf{d} \tag{17}$$

$$\mathbf{d}_1 = a_1(\mathbf{W}_{d1}^T \mathbf{d}_0 + \mathbf{b}_{d1}), \tag{18}$$

$$\ldots\ldots \tag{19}$$

$$\mathbf{d}_{L-1} = a_{L-1}(\mathbf{W}_{d(L-1)}^T \mathbf{d}_{L-2} + \mathbf{b}_{d(L-1)}), \tag{20}$$

$$f_{deep}(\mathbf{d}_c, \mathbf{d}_x|\Theta_d) = a_L(\mathbf{W}_{dL}^T \mathbf{d}_{L-1} + \mathbf{b}_{dL}), \tag{21}$$

where $\{\mathbf{W}_{d1}, \ldots, \mathbf{W}_{dL}\}$ and $\{\mathbf{b}_{d1}, \ldots, \mathbf{b}_{dL}\}$ denote the weight matrices and the bias vectors for the deep model, and $\Theta_d = \{\mathbf{W}_{d1},\ldots,\mathbf{W}_{dL}, \mathbf{b}_{d1},\ldots,\mathbf{b}_{dL}\}$ denotes the entire set of parameters in the deep model.

4.5 Training the Network

Now we discuss learning the wide-and-deep network parameters Θ. 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 $\mathbb{V}_i$ and we also include a sampled set of negative visualizations $\widehat{\mathbb{V}}_i^-$ (as in Def. 6). The set of training visualizations for $\mathbf{X}_i$ is then $\mathbb{V}_i \cup \widehat{\mathbb{V}}_i^-$. During training, each visualization $\mathcal{V}_{ik}$ comes from an arbitrary dataset $\mathbf{X}_i$ and has a binary ground-truth label $Y_{ik} \in \{0, 1\}$. 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 for all visualizations in the entire corpus $\mathcal{D}$ consisting of N datasets $\{\mathbf{X}_i\}_{i=1}^N$. Eq. 22 shows the likelihood: for each dataset $\mathbf{X}_i$ we have the set $\mathbb{V}_i \cup \widehat{\mathbb{V}}_i^- = \{\ldots, (\mathbf{X}_i^{(k)}, C_{ik}), \ldots\}$ of training visualizations where each visualization $(\mathbf{X}_i^{(k)}, C_{ik}) \in \mathbb{V}_i \cup \widehat{\mathbb{V}}_i^-$ consists of a configuration $C_{ik} \in \mathcal{C}$ and a subset of attributes $\mathbf{X}_i^{(k)}$ from dataset $\mathbf{X}_i$.

$$p(\widehat{\mathbb{V}}_i^-, \mathbb{V}_i|\Theta) = \prod_{(\mathbf{X}_i^{(k)}, C_{ik}) \in \mathbb{V}_i} \hat{Y}_{ik} \prod_{(\mathbf{X}_i^{(k)}, C_{ik}) \in \widehat{\mathbb{V}}_i^-} \left(1 - \hat{Y}_{ik}\right), \quad \text{for } i = 1, \ldots, N \tag{22}$$

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. 22 and summing over all

datasets $\{\mathbf{X}_i\}_{i=1}^N$ give us the loss L.

$$L = \sum_{i=1}^{N} \Big(- \sum_{(\mathbf{X}_i^{(k)}, C_{ik}) \in \mathbb{V}_i} \log \hat{Y}_{ik} - \sum_{(\mathbf{X}_i^{(k)}, C_{ik}) \in \widehat{\mathbb{V}}_i^-} \log(1 - \hat{Y}_{ik}) \Big)$$
$$= - \sum_{i=1}^{N} \sum_{(\mathbf{X}_i^{(k)}, C_{ik}) \in \mathbb{V}_i \cup \widehat{\mathbb{V}}_i^-} Y_{ik} \log \hat{Y}_{ik} + (1 - Y_{ik}) \log(1 - \hat{Y}_{ik}) \tag{23}$$

We minimize the objective function through gradient-based optimization to update the model parameters Θ in $\mathcal{M}$.

4.6 Inference

Given the *trained* wide-and-deep visualization recommendation model $\mathcal{M}$ from Sec. 4.5, we now describe the inference procedure for scoring and recommending visualizations for an arbitrary new dataset of interest. As illustrated in the lower part of Figure 2, given an arbitrary dataset $\mathbf{X}_{\text{test}}$ selected or uploaded by an arbitrary user, we generate the space of visualizations $\mathbb{V}^{\star}_{\text{test}} = \{\ldots, \mathcal{V}^{(k)}_{\text{test}}, \ldots\}$ through Def. 3 process, where each visualization $\mathcal{V}^{(k)}_{\text{test}}$ consists of a subset of attributes $\mathbf{X}^{(k)}_{\text{test}}$ from the dataset $\mathbf{X}_{\text{test}}$ and a configuration $C \in \mathcal{C}$, *i.e.*, $\mathcal{V}^{(k)}_{\text{test}} = (\mathbf{X}^{(k)}_{\text{test}}, C)$.

Each visualization $\mathcal{V}^{(k)}_{\text{test}}$ is 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 $\mathbf{X}^{(k)}_{\text{test}}$, we derive the meta-feature $\mathbf{d}_{x_i}$ for each attribute $\mathbf{x}_i \in \mathbf{X}^{(k)}_{\text{test}}$. Depending on the attributes used in the visualizations, the meta-features get concatenated to get an overall dense feature vector $\mathbf{d}_x$. To obtain the sparse feature vector $\mathbf{s}_x$ for the selected attributes, we bin $\mathbf{d}_x$. 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. We then weight them by w_{wide} and w_{deep} to get a final score $\hat{Y}_{test,k} \in [0, 1]$ as follows,

$$\hat{Y}_{test,k} = f(\mathbf{X}^{(k)}_{\text{test}}, C|\Theta) \tag{24}$$
$$= \sigma(w_{wide} f_{wide}(\mathbf{s}_c, \mathbf{s}_x|\Theta_s) + w_{deep} f_{deep}(\mathbf{d}_c, \mathbf{d}_x|\Theta_d))$$

We repeat the above to score all possible visualizations in $\mathbb{V}^{\star}_{\text{test}}$, and then recommend the top visualizations based on the predicted scores. This process is consistent with Def. 9. We expand more details about the evaluation of this process in Sec. A.3.

5 EXPERIMENTS

We conduct experiments to answer the following questions:

- **RQ1:** Can our model recover previously known "top" visualizations for unseen user-selected datasets, and do this better than reasonable baselines (Sec. 5.1)?
- **RQ2:** Do human experts prefer our ML-based visualization recommendations to a state-of-the-art rule-based system (Sec. 5.2)?
- **RQ3:** How do our recommendations differ from a rule-based baseline in qualitative terms (Sec. 5.3)?

Table 2: Quantitative Results for Visualization Recommendation. See text for discussion.

	nDCG′ (Eq. 25)					
Model	@1	@2	@5	@10	@20	Rank
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-MLP (deep-only)	0.804	0.807	0.851	0.866	0.887	2
Ours-GMF (wide-only)	0.721	0.714	0.768	0.801	0.839	3
Ours	**0.827**	**0.827**	**0.867**	**0.882**	**0.897**	1

5.1 Quantitative Results

We answer RQ1 by quantitatively evaluating the ranking of visualizations from our ML-based visualization recommendation model using the evaluation framework from Sec. A. For training ML-based models, we create a training corpus randomly sampled from the Plot.ly community feed.[6] The training corpus consisted of $N = 925$ datasets, $\sum_{i=1}^{N} |\mathbf{X}_i| = 11,778$ attributes, and $\sum_{i=1}^{N} |\mathbb{V}_i| = 4,865$ positive (relevant) visualizations across all N datasets. Further, each dataset had an average of $\sum_{i=1}^{N} |\mathbf{x}_i|/N = 12.73$ attributes and 5.25 positive visualizations. The held-out test corpus consisted of 232 positive (previously known "top") visualizations across 46 test datasets (sampled uniformly at random), which had 299 attributes. Results are averaged over 10 trials.

We compare our approach against two common-sense baselines along with two stronger baselines, which are simpler variants of the proposed ML-based model. The first baseline, *random*, recommends the top-k visualizations chosen uniformly at random from the set of positive and sampled negative visualizations for a specific dataset. The second baseline, *ConfigPop*, scores visualizations based on the popularity of a visualization configuration. We also include two stronger baselines, including an MLP baseline that uses only the deep component of our model, along with the generalized matrix factorization (GMF) baseline that uses only the wide component.

To evaluate the effectiveness of the top recommended visualizations, we use the modified normalized Discounted Cumulative Gain (nDCG) proposed in Eq. 25 at $k \in \{1, 2, 5, 10, 20\}$ for the different top-k visualization recommendations (nDCG@k). Results are reported in Table 2. Strikingly, our wide-and-deep learning-based model performs the best, achieving a high nDCG across all $k = 1, ..., 20$, as shown in Table 2. 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). Furthermore, our approach with both the wide and deep components, has the highest nDCG scores across all $k \in \{1, 2, 5, 10, 20\}$ 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 2 (RQ1). Note that results at smaller k are 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 $k = 1$. This result clearly

[6] Our dataset is available at https://github.com/xeniaqian94/kdd21-MLVis.

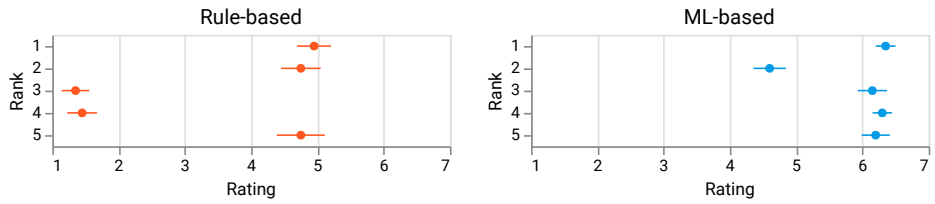


Figure 3: Averaged human experts' ratings on top 5 visualizations from rule-based (Voyager2 using CompassQL) and our ML-based system. Nearly all visualizations from the ML-based system received higher ratings. Strikingly, the top-4 visualizations receiving the highest rating by human experts are those from the ML-based system.

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 visualizations that were generated by an actual user. These findings confirm that our model learns to recommend high quality visualizations that a user finds most relevant from arbitrary unseen datasets.

5.2 User Study

In this section, we perform a user study to compare our end-to-end ML-based visualization recommender system to an existing state-of-the-art rule-based system (Voyager2 with CompassQL).

5.2.1 Methods. We recruited 20 participants (12 males and 8 females, aged 20–40) by sending out an email to a mailing list of visualization experts within an industry research division and a public research university. Participants rated top 5 visualization recommendations from both our ML-based approach and the rule-based approach (10 visualizations in total) in a within-subjects experiment design. The top 5 recommendations for each approach were computed from the standard `cars` dataset, which consists of 398 observations (rows, cars) along with 10 attributes (columns) including three categorical, six numerical, and one temporal attribute. The 10 visualizations were presented in a shuffled order that was consistent across participants, with no information about the recommendation source. Participants were asked to rate the *quality* of each visualization using a 7-point Likert scale (1 = Very Poor, 4 = Fair, 7 = Very Good).

5.2.2 Results. Participants judged the ML-recommended visualizations ($M = 5.92, SD = 1.09, MED = 6$) to be of significantly higher quality compared to the rule-recommended visualizations ($M = 3.45, SD = 2.07, MED = 4$). Strikingly, the top ranked visualization (top-1) is exactly the top ranked visualization from our ML-based system. Furthermore, among the 10 visualizations, the top 4 visualizations with the highest score come from our ML-based visualization recommendation system, and not from the rule-based system. As shown in Figure 3, the variance of the ML-based recommendations are nearly always less than the rule-based system.

5.3 Qualitative Analysis

To investigate why our ML-based approach might be more effective than a rule-based approach, including whether it can automatically learn general rules that are preferred by experts (RQ3), we conducted a qualitative case study. We specified a number of queries for visualizations from the `cars` dataset, using a simple user-interface with our recommendation system (Figure 7).

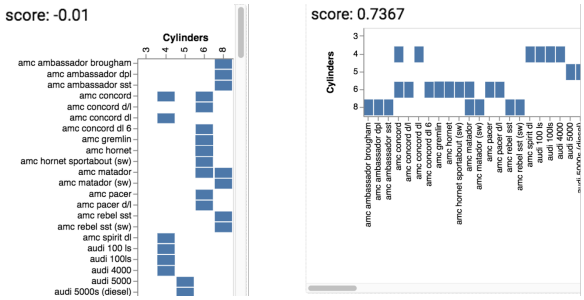


Figure 4: Top visualizations from rule-based (left) vs. our ML-based approach (right). The ML-based approach learns to recommend a horizontal layout and penalizes the vertical charts, which is consistent with domain experts.

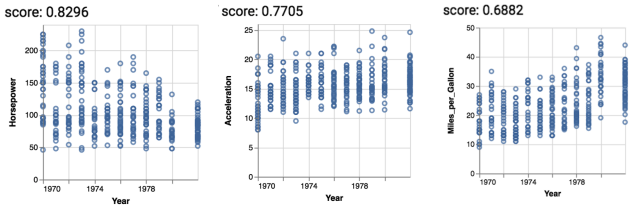
5.3.1 Learning to place attributes like an expert. Using a query for visualizations containing two nominal attributes, we see that both approaches recommend visualizations with the attribute car *model name* in their top visualizations as shown in Figure 4. However, the rule-based system incorrectly 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 model penalizes the vertical charts to ensure they are not recommended to the user.

5.3.2 Scoring and ranking issues of rule-based systems. The over-simplified scoring used by existing rule-based systems results in many visualizations having exactly the same score, and therefore, no way to actually rank them. In Figure 5, we show one such case where the top visualizations from the rule-based are all assigned the same score of 0. As an aside, there are even more visualizations down the list with a score of 0. Because of this lack of resolution, the rule-based system may rank low quality visualizations higher than some visualizations that are clearly better. In comparison, our ML-based approach infers more meaningful scores to appropriately rank the visualizations by relevance.

6 CONCLUSION

In this work, we proposed the first end-to-end ML-based visualization recommendation system. We first formalized the problem and described a generic learning framework for solving it. Next, we proposed a wide-and-deep learning approach 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

(a) Top 3 Rule-based Vis. Rec. (CompassQL/Voyager2)



(b) Top 3 ML-based Vis. Rec. (Ours)

Figure 5: Rule-based scoring and ranking issues vs. our ML-based approach that infers more meaningful scores to appropriately rank the visualizations by relevance.

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. We demonstrate the effectiveness of our ML-based visualization recommendation system through comprehensive experiments, namely, a quantitative evaluation of the visualization recommendations using known ground-truth, a user study with 20 human experts that show a clear preference for visualizations recommended by the ML-based model.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE* 17, 6 (2005), 734–749.

[2] Anushka Anand and Justin Talbot. 2015. Automatic selection of partitioning variables for small multiple displays. *TVCG* 22, 1 (2015), 669–677.

[3] James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *KDD Cup*. 35.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Workshop on Deep Learning for Recommender Systems*. 7–10.

[5] Tuan Nhon Dang and Leland Wilkinson. 2014. Scagexplorer: Exploring scatterplots by their scagnostics. In *IEEE Pacific visualization symposium*.

[6] Cagatay Demiralp, Peter J Haas, Srinivasan Parthasarathy, and Tejaswini Pedapati. 2017. Foresight: Recommending Visual Insights. *VLDB Endowment* (2017).

[7] Mark Derthick, John Kolojejchick, and Steven F Roth. 1997. An interactive visualization environment for data exploration. In *KDD*. 2–9.

[8] Victor Dibia and Çağatay Demiralp. 2019. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE computer graphics and applications* 39, 5 (2019), 33–46.

[9] Humaira Ehsan, Mohamed Sharaf, and Panos Chrysanthis. 2016. Muve: Efficient multi-objective view recommendation for visual data exploration. In *ICDE*.

[10] David Gotz and Zhen Wen. 2009. Behavior-driven visualization recommendation. In *IUI*. 315–324.

[11] Camille Harris, Ryan A Rossi, Sana Malik, Jane Hoffswell, Fan Du, Tak Yeon Lee, Eunyee Koh, and Handong Zhao. 2021. Insight-centric Visualization Recommendation. *arXiv:2103.11297* (2021).

[12] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*. 355–364.

[13] Jeffrey Heer, Nicholas Kong, and Maneesh Agrawala. 2009. Sizing the horizon: the effects of chart size and layering on the graphical perception of time series visualizations. In *CHI*. 1303–1312.

[14] Kevin Hu, Michiel Bakker, Stephen Li, Tim Kraska, and César Hidalgo. 2019. Vizml: A machine learning approach to visualization recommendation. In *CHI*.

[15] Kevin Hu, Diana Orghian, and César Hidalgo. 2018. Dive: A mixed-initiative system supporting integrated data exploration workflows. In *Workshop on Human-In-the-Loop Data Anal.* 1–7.

[16] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *TOIS* 20, 4 (2002), 422–446.

[17] Younghoon Kim and Jeffrey Heer. 2018. Assessing Effects of Task and Data Distribution on the Effectiveness of Visual Encodings. *EuroVis* (2018).

[18] Doris Jung-Lin Lee. 2020. *Insight Machines: The Past, Present, and Future of Visualization Recommendation.*

[19] Doris Jung-Lin Lee, Himel Dev, Huizi Hu, Hazem Elmeleegy, and Aditya Parameswaran. 2019. Avoiding drill-down fallacies with VisPilot: assisted exploration of data subsets. In *IUI*. 186–196.

[20] Doris Jung-Lin Lee, John Lee, Tarique Siddiqui, Jaewoo Kim, Karrie Karahalios, and Aditya Parameswaran. 2019. You can't always sketch what you want: Understanding Sensemaking in Visual Query Systems. *TVCG* 26, 1 (2019), 1267–1277.

[21] Halden Lin, Dominik Moritz, and Jeffrey Heer. 2020. Dziban: Balancing Agency & Automation in Visualization Design via Anchored Recommendations. In *CHI*.

[22] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing* 7, 1 (2003), 76–80.

[23] Yang Liu and Jeffrey Heer. 2018. Somewhere over the rainbow: An empirical assessment of quantitative colormaps. In *CHI*. 1–12.

[24] Jock Mackinlay. 1986. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.* 5, 2 (1986), 110–141.

[25] Jock Mackinlay, Pat Hanrahan, and Chris Stolte. 2007. Show Me: Automatic presentation for visual analysis. *TVCG* 13, 6 (2007), 1137–1144.

[26] 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. *TVCG* 25, 1 (2018).

[27] Michael Oppermann, Robert Kincaid, and Tamara Munzner. 2021. VizCommender: Computing Text-Based Similarity in Visualization Repositories for Content-Based Recommendations. *TVCG* 27, 02 (2021), 495–505.

[28] Daniel B Perry, Bill Howe, Alicia MF Key, and Cecilia Aragon. 2013. VizDeck: Streamlining exploratory visual analytics of scientific data. In *iConference*.

[29] Xin Qian, Ryan A Rossi, Fan Du, Sungchul Kim, Eunyee Koh, Sana Malik, Tak Yeon Lee, and Nesreen K Ahmed. 2021. Personalized Visualization Recommendation. *arXiv:2102.06343* (2021).

[30] Xin Qian, Ryan A Rossi, Fan Du, Sungchul Kim, Eunyee Koh, Sana Malik, Tak Yeon Lee, and Joel Chan. 2020. ML-based Visualization Recommendation: Learning to Recommend Visualizations from Data. *arXiv:2009.12316* (2020).

[31] Steven F Roth, John Kolojejchick, Joe Mattis, and Jade Goldstein. 1994. Interactive graphic design using automatic presentation knowledge. In *CHI*. 112–117.

[32] Bahador Saket, Alex Endert, and Çağatay Demiralp. 2018. Task-based effectiveness of basic visualizations. *TVCG* 25, 7 (2018), 2505–2512.

[33] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. 1998. Discovery-driven exploration of OLAP data cubes. In *Extending Database Tech.* 168–182.

[34] Marc M. Sebrechts, John V. Cugini, Sharon J. Laskowski, Joanna Vasilakis, and Michael S. Miller. 1999. Visualization of Search Results: A Comparative Evaluation of Text, 2D, and 3D Interfaces. In *SIGIR*.

[35] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. 2016. Effortless Data Exploration with zenvisage: An Expressive and Interactive Visual Analytics System. *VLDB Endowment* (2016).

[36] Chris Stolte, Diane Tang, and Pat Hanrahan. 2002. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *TVCG* 8, 1 (2002), 52–65.

[37] Manasi Vartak, Silu Huang, Tarique Siddiqui, Samuel Madden, and Aditya Parameswaran. 2017. Towards visualization recommendation systems. *SIGMOD* 45, 4 (2017), 34–39.

[38] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. 2015. SeeDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics. (2015).

[39] Leland Wilkinson, Anushka Anand, and Robert Grossman. 2005. Graph-theoretic scagnostics. In *IEEE Symposium on Information Visualization*. 157–164.

[40] Graham Wills and Leland Wilkinson. 2010. Autovis: automatic visualization. *Information Visualization* 9, 1 (2010), 47–69.

[41] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2015. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *TVCG* 22, 1 (2015), 649–658.

[42] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Towards a general-purpose query language for visualization recommendation. In *Workshop on Human-In-the-Loop Data Anal.*

[43] 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*. 2648–2659.

[44] Aoyu Wu, Liwenhan Xie, Bongshin Lee, Yun Wang, Weiwei Cui, and Huamin Qu. 2021. Learning to Automate Chart Layout Configurations Using Crowdsourced Paired Comparison. In *CHI*.

Appendix

Table 3: Summary of meta-feature functions for an attribute.

The functions are called from the learning framework in Table 1. Let $\mathbf{x}$ *denote an arbitrary attribute vector and* $\pi(\mathbf{x})$ *is the sorted vector of* $\mathbf{x}$.

Name	Equation
Num. values in attribute	$\lvert\mathbf{x}\rvert$
Num./frac. missing values	s, $^{(\lvert\mathbf{x}\rvert-s)}/_{\lvert\mathbf{x}\rvert}$
Num. nonzeros, density	$\mathsf{nnz}(\mathbf{x})$, $^{\mathsf{nnz}(\mathbf{x})}/_{\lvert\mathbf{x}\rvert}$
Num. unique values	$\mathsf{card}(\mathbf{x})$
Q_1, Q_3	median of $\lvert\mathbf{x}\rvert/2$ 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 $\alpha \in \{1.5, 3\}$	$\sum_i \mathbb{I}(x_i < Q_1 - \alpha IQR) + \sum_i \mathbb{I}(x_i > Q_3 + \alpha IQR)$
(αstd) outliers $\alpha \in \{2, 3\}$	$\mu_\mathbf{x} \pm \alpha\sigma_\mathbf{x}$
Spearman (ρ, p-val)	$\mathsf{spearman}(\mathbf{x}, \pi(\mathbf{x}))$
Kendall (τ, p-val)	$\mathsf{kendall}(\mathbf{x}, \pi(\mathbf{x}))$
Pearson (r, p-val)	$\mathsf{pearson}(\mathbf{x}, \pi(\mathbf{x}))$
Min, max, range, median	$\min(\mathbf{x})$, $\max(\mathbf{x})$, $\max(\mathbf{x}) - \min(\mathbf{x})$, $\mathsf{med}(\mathbf{x})$
Geometric Mean	$\lvert\mathbf{x}\rvert^{-1} \prod_i x_i$
Harmonic Mean	$\lvert\mathbf{x}\rvert / \sum_i \frac{1}{x_i}$
Mean, Stdev, Variance	$\mu_\mathbf{x}$, $\sigma_\mathbf{x}$, $\sigma^2_\mathbf{x}$
Skewness, Kurtosis	$^{\mathbb{E}(\mathbf{x}-\mu_\mathbf{x})^3}/_{\sigma^3_\mathbf{x}}$, $^{\mathbb{E}(\mathbf{x}-\mu_\mathbf{x})^4}/_{\sigma^4_\mathbf{x}}$
HyperSkewness	$^{\mathbb{E}(\mathbf{x}-\mu_\mathbf{x})^5}/_{\sigma^5_\mathbf{x}}$
Moments[6–10], k-stat.[3–4]	–
Quartile Dispersion Coeff.	$\frac{Q_3-Q_1}{Q_3+Q_1}$
Median Absolute Deviation	$\mathsf{med}(\lvert\mathbf{x} - \mathsf{med}(\mathbf{x})\rvert)$
Avg. Absolute Deviation	$\frac{1}{\lvert\mathbf{x}\rvert}\mathbf{e}^T \lvert\mathbf{x} - \mu_\mathbf{x}\rvert$
Coeff. of Variation	$^{\sigma_\mathbf{x}}/_{\mu_\mathbf{x}}$
Efficiency ratio	$^{\sigma^2_\mathbf{x}}/_{\mu^2_\mathbf{x}}$
Variance-to-mean ratio	$^{\sigma^2_\mathbf{x}}/_{\mu_\mathbf{x}}$
Signal-to-noise ratio (SNR)	$^{\mu^2_\mathbf{x}}/_{\sigma^2_\mathbf{x}}$
Entropy, Norm. entropy	$H(\mathbf{x}) = -\sum_i x_i \log x_i$, $^{H(\mathbf{x})}/_{\log_2\lvert\mathbf{x}\rvert}$
Gini coefficient	–
Quartile max gap	$\max(Q_{i+1} - Q_i)$
Centroid max gap	$\max_{ij} \lvert c_i - c_j\rvert$
Histogram prob. dist.	$\mathbf{p}_h = \frac{\mathbf{h}}{\mathbf{h}^T\mathbf{e}}$ (with fixed # of bins)

A EVALUATION FRAMEWORK

Here we describe a supporting evaluation framework for end-to-end *ML-based* visualization recommendation (Sec. 3). Four differences between our problem and traditional item-based recommendation prevent us from leveraging commonly used evaluation techniques:

- **Visualization complexity (Sec. A.1)**: While traditional recommender systems score simple objects such as an item, ML-based visualization recommendation models must learn from visualizations that are more complex objects (Def. 3).
- **No shared recommendation space (Sec. A.1)**: 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. A.2–A.3)**: 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.
- **Dynamic & dataset dependent vis. space (Sec. A.2–A.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.

Our proposed framework—comprising specifications of the corpus of datasets and visualizations, construction of training and testing sets, and evaluation metrics—is designed to overcome challenges from these differences, and serves as a fundamental basis for systematically evaluating ML-based visualization recommender systems, including our own model and those that arise in the future.

A.1 Corpus: Datasets and Visualizations

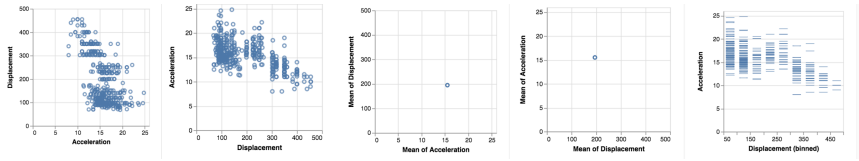
In traditional recommender systems that recommend items to users [1], there is a *single shared set of items* (*e.g.*, movies [3], products [22], etc.). However, in visualization recommendation, there is no shared set of visualizations to recommend to users, as it depends entirely on the dataset of interest. Hence, if we have N datasets, then there are N completely disjoint sets of visualizations that can be recommended. 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, and some attributes in the dataset may never be used in a visualization. Thus, the goal is to learn a model to score and ultimately recommend visualizations generated from any unseen dataset in the future. Every new dataset gives rise to an exponential amount of possible visualizations that one must search. This makes this recommendation problem extremely challenging. In addition, the visualization search space is also completely disjoint from the search space of another arbitrary dataset as shown in Lemma 1.

A.2 Training from the Corpus

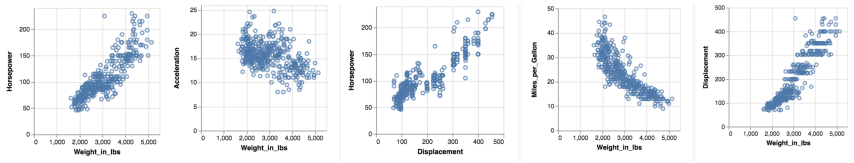
The next step is to create a training set from the corpus of datasets and visualizations. For example, the wide-and-deep network approach addresses the issue of the dynamic space of visualizations by creating visualizations that come from a combination of attributes and a visualization configuration. 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 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 non-relevant instances do 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, every dataset has a unique 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. However, other ways of sampling non-relevant visualizations can also be used.

A.3 Testing and Deployment

Now, we discuss how a trained visualization recommendation model $\mathcal{M}$ can be used for testing and deployment. Given a new or selected



(a) Top 5 Rule-based Vis Rec.



(b) Top 5 ML-based Vis. Rec. (Ours)

Figure 6: 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.

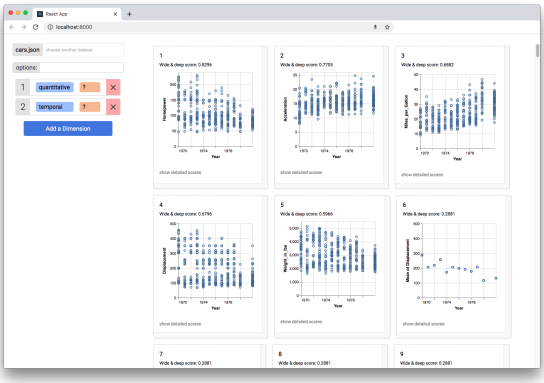


Figure 7: Screenshot of our end-to-end ML-based visualization recommender system.

held-out dataset $\mathbf{X}_{\text{test}}$, 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) [1, 3, 22], our framework generates a ranking of visualizations to recommend with on-the-fly, which are dependent on the new unseen dataset. When experimenting with different models, we did negative sampling of non-relevant visualizations, which allows us to efficiently report an average result from more datasets. We describe the evaluation metric for our visualization recommendation problem in Sec. A.4. When the model gets deployed and tested on a real-world new dataset, the recommended visualizations are selected from the entire space of visualizations for that dataset (as in Def. 3).

A.4 Evaluation Metrics

Evaluating the quality of the ranking of visualizations (recommendations) given by the learned model has unique challenges. As mentioned earlier, 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 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 $\mathbf{X}_i$. Thus, standard ranking metrics that assume a fixed space of objects to rank over such as nDCG [16] cannot be used directly for visualization recommendation. Furthermore, the number of possible visualizations the ranking is computed over depends entirely on the dataset and 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 it. Taking a random model for instance, it is easy to obtain a high nDCG when a dataset has only two attributes as opposed to hundreds.

The nuances drive us to further specify nDCG for evaluating 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 $\{\mathbb{V}_i\}_{i=1}^{N}$, the corrected nDCG is defined as,

$$nDCG@K = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{Z_i^K}\sum_{j=1}^{K}\frac{2^{Y_{ij}}-1}{log_2(j+1)} \tag{25}$$

$$Z_i^K = \sum_{j=1}^{\min(K,|\mathbb{V}_i|)}\frac{1}{log_2(j+1)} \tag{26}$$

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 $\mathbf{X}_i$, and Z_i^K is the dataset-dependent normalization factor that ensures a perfect visualization ranking for dataset $\mathbf{X}_i$ receives a perfect score of 1. This is required since each dataset $\mathbf{X}_i$ almost surely has a different number of positive visualizations $|\mathbb{V}_i|$, that is, for any arbitrary two datasets $\mathbf{X}_i$ and $\mathbf{X}_j$, $|\mathbb{V}_i| \neq |\mathbb{V}_j|$. 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 $\mathbf{X}_i$ since $1/\log 2(j+2)$ decreases quickly and then asymptotes to a constant as j increases.

A.5 Implementation and Model Configuration

For training, each positive (relevant) visualizations is complemented with four negative visualizations, sampled uniformly at random. Each test dataset has 99 negative visualizations. The wide-and-deep approach is implemented in PyTorch v.1.3.0. A dense meta-feature vector of an attribute is 1,006-dim. Configuration embeddings are 128-dim. Deriving sparse feature vectors $\mathbf{s}_x$ uses a bin size of five. The deep model uses a 3-layer MLP (with ReLu activation). Optimization uses the BCE loss, the Adam optimizer (lr = $1e^{-3}$) and a batch size of four. Training ends after three epoches.