DIGITAL ACCESS TO SCHOLARSHIP AT HARVARD
DASH.HARVARD.EDU

HARVARD LIBRARY
Office for Scholarly Communication

Analyzing and Evaluating Post hoc Explanation Methods for Black Box Machine Learning

Citation
Pombra, Javin. 2022. Analyzing and Evaluating Post hoc Explanation Methods for Black Box Machine Learning. Bachelor's thesis, Harvard College.

Permanent link
https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37371734

Terms of Use
This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

Share Your Story

The Harvard community has made this article openly available.
Please share how this access benefits you. Submit a story .

Accessibility

Analyzing and Evaluating Post hoc Explanation Methods for Black Box Machine Learning

A DISSERTATION PRESENTED
BY
JAVIN POMBRA
TO
THE DEPARTMENT OF COMPUTER SCIENCE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
HONOR'S BACHELOR OF ARTS
IN THE SUBJECT OF
COMPUTER SCIENCE

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
MAY 2022

Thesis advisor: Professor Himabindu Lakkaraju Javin Pombra

Analyzing and Evaluating Post hoc Explanation Methods for Black Box Machine Learning

Abstract

Over the past decade, complex tools such as deep learning models have been increasingly employed in high-stakes domains such as healthcare and criminal justice. Furthermore, these models achieve state-of-the-art accuracy at the expense of interpretability. As a result, practitioners, end users, and regulators have expressed a strong desire to increase the availability of post hoc explanation methods or ways to explain complex model architectures after the model is trained and deployed. Unfortunately, given the nascence of the field of explainability, there is little to no work on comparing and analyzing the behavior of popular post hoc methods.

This work introduces the disagreement problem in explainable machine learning. Through a series of user studies and offline experiments, we establish that the most common post hoc methods deployed on tabular, vision, and language datasets exhibit significant disagreement. Once established, we aim then to resolve the disagreement problem within graph neural network and deep learning recommendation models. To this end, we formalize novel metrics to test the efficacy of explainability methods. Starting with evaluating ex-

plainability for graph neural networks, we show under what dataset and model conditions various post hoc explainability methods operate best. We then move to the recommendation modeling space, formulating explainability as a joint task of interpreting embedding layers and neural layers. In addition to presenting a novel method, we conduct offline and online experimentation to also present which methods are preferred by target users.

Contents

Dedicated to Sandeep, Jasmeen, and Karnika Pombra.

0

Introduction

Machine learning's widespread use has combined with the fast-paced nature of data collection and big data practices. That is, due to the readily available data present today in vision, healthcare, business, criminal justice, and other domains, complex models have become the norm. Due to the large amount of data they process, these models have also become increasingly black box. Practitioners are long past the days of linear regression and interepretable,

small decision trees. Today, the most impressive accuracy results come from the millions of unintelligible weights of a neural network.

Due to their impressive test accuracy, many of these models are increasingly being deployed in high-stake domains. Whether it is setting the bail of those accused or deciding priority in treatment for various patients with diabetes, models are being used in ways that have an incredible impact on the livelihood of humans. And, because of the complexity of these models, it is often impossible to understand the model simply from its parameters. Instead, several techniques have been proposed in recent literature to explain complex models in a post hoc fashion.

Most of the popular methods focus on explaining individual predictions (i.e., local explanations) of any given model and can be broadly categorized into local or global methods along with model-agnostic or model-dependent methods. There have been a variety and somewhat disorganized attempts to quantify the notion of explainability. While these metrics have attempted to analyze the correctness of explanations, a lack of ground truth often makes it difficult to agree on the notion of correctness of explanations. Owing in part to the fact that explainability has diverse definitions in the machine learning community, the metrics to measure the same are also diverse and disparate.

As a result of disparate metrics for explainability, there are no clear guidelines on which state-of-the-art explainability methods to use in practice. For instance, practitioners do not

typically rely on a single explanation method, but instead employ multiple such methods simultaneously to understand the underlying models. This approach however is predicated on consistent outputs from the most commonly used post hoc methods. However, popular post hoc explanation methods such as LIME and SHAP output different explanations (or feature attributions). This is a prime example of the **disagreement problem** in explainable machine learning.

Given all of the above, it is critical not only to understand and quantify how often explanations output by state-of-the-art methods disagree with each other, but also to study how such disagreements are currently being resolved by machine learning practitioners. We hypothesize that state-of-the-art disagreement methods disagree and will test this hypothesis in chapter 3 of this thesis.

Armed with the knowledge that machine learning explainability methods have nontrivial, and often significant, areas of disagreement, we begin the arduous but important work of resolving that disagreement. We approach this by resolving disagreement in explainability for the model class of graph neural networks and for the application area of recommendation modeling. Our choice in model class and application area come from looking at machine learning areas that have received considerably less attention when it comes to post hoc explainability.

We begin the task of resolving the disagreement by focusing on **graph neural networks.**

In recent years, graph neural networks (GNNs) have demonstrated breakthroughs in numerous domains that involve complex graph structures (e.g., recommender systems, drug discovery, NLP, etc.)[13,26,59]. However, due to their message-passing mechanisms, GNNs lack transparency in predictions. In order to make these black box models more transparent, numerous GNN explainability methods have been proposed that tackle node feature, node, and edge attribution[63]. Explainability for GNNs is particularly important given practitioner's hesitancy to utilize models and the often human-centric problems that GNNs tackle.

In the face of a wide array of explainability techniques, use cases, and deployment environments, there is little to no standardization in evaluation. To date, no universal set of quantitative metrics has been proposed that not only evaluates the explainability method's performance, but also accounts for critical real-world deployment objectives such as fairness and scalability. In this work, we identify, define, and compute a set of method-agnostic metrics that can be used to evaluate explainability methods along these desiderata.

Another challenge unaddressed by current literature is the evaluation of these metrics in the context of use case. The explanation results heavily depend on the input GNN model, the dataset structure, and explanation technique chosen, and thus must be compared using these factors as a frame of reference. To the best of our knowledge, this will be the first work to explore the interaction between explainability method performance and the GNN

model and dataset it was trained upon. It will also be the first work which analyzes when to use a variety of explainability methods in what situations (i.e. types of datasets or models).

We then move to resolving the disagreement problem in the recommendation modeling space. Recommendation models are one of the most consistent, significant, and wide-reaching applications of modern machine and deep learning. A combination of new devices to collect data and wide-spread use of information has lead to an exponential growth of data available on the web in spheres of commerce, healthcare, and more. As a result, users must choose between more products such as restaurants, movies, and travel locations than ever before. In order to better equip users to make these decisions, personalization is more important than ever which has meant recommendation engines have become pervasive across web domains such as social media and e-commerce. For example, 80 % of movies watched on Netflix came from recommendations[45], and 60 % of video clicks came from home page recommendation in YouTube[9]. Moreover, 35 % of what customers purchase on Amazon comes from product recommendations and at Airbnb, Search Ranking and Similar Listings drive 99 % of all booking conversions[32].

Recently, many companies have employed deep learning to further improve their recommendation quality. Developers at Youtube have presented a deep neural network-based recommendation algorithm for video recommendations.[9]. Cheng et al. proposed an App recommender system for Google Play with a wide and deep model[6]. Shumpei et al. pre-

sented an RNN-based news recommender system for Yahoo! News[2]. In both offline metrics such as accuracy and online metrics such as impressions and clicks, these methods have shown significant improvement over traditional collaborative and content filtering models. Furthermore, the number of publications on deep learning-based recommendation methods has increased exponentially recently since the mid 2010's, providing evidence that the community both academically and in industry has moved away from traditional models. The leading international conference on recommender system, RecSys, began organizing regular workshop on deep learning for recommender systems in 2016. Today, there is over 40 trillion gigabytes of data generated each day, which consists of social media data, emails, and internet searches, meaning only a greater drive towards deep learning systems[1].

Explainability in recommender systems has never been more important. First, recent studies have shown that recommender systems have the possibility of having harmful nudges in social media contexts. For example, whether it is political polarization and radicalization or causing deep mental health problems in at-risk populations such as adolescents, understanding why recommendations are made is a necessity[7]. Second, a variety of national and international regulators have moved towards mandating explainability in machine learning models utilized. For instance, the European Union's General Data Protection Regulations for AI includes language mandating the transparency of artificial intelligence[54]. With knowledge of these benefits, practitioners must employ post hoc methods. However, given

the existence of the disagreement problem, we understand that these methods need to be studied further.

Ultimately, this thesis hopes to be the start of and the inspiration for more work in the disagreement space. If our state-of-the-art machine learning models constantly disagreed on a prediction, we would immediately understand the severity of the situation. As the current machine learning community move towards using more and more complex data, resolving this problem is the first step. A complete list of our contributions split into the three sections of this thesis.

The first contribution of this thesis will be to formalize the disagreement problem. This work was done in collabration with the AI4LIFE lab. The specific contributions were as follows:

1. We first obtain practitioner inputs on what constitutes explanation disagreement, and the extent to which they encounter this problem in their day-to-day workflow. To this end, we conduct semi-structured interviews with data scientists (N = 25) who regularly work with explainability tools.
2. Using the insights obtained from the aforementioned interviews, we formalize the notion of explanation disagreement, and propose a novel evaluation framework which can quantitatively measure the disagreement between any two explanations that explain the same model prediction.
3. We leverage the aforementioned framework to carry out a rigorous empirical analysis with real-world data to quantify the level of disagreement between popular post hoc explanation methods. To this end, we experiment with four real-world datasets spanning three data modalities?, six state-of-the-art explanation methods, and various popular predictive models (e.g., logistic regression, tree based models, deep neural

networks, recurrent neural networks such as LSTM models, and convolutional neural networks such as ResNet).

4. Lastly, we study how explanation disagreements are **currently** resolved in practice.. To this end, we carry out an online user study with data scientists (N = 24) where we show them pairs of explanations that disagree with each other, and ask them which explanation (if any) would they rely on and why. At the end of this survey, we also ask our study participants to provide a high-level description of the strategies they use to resolve explanation disagreements in their day-to-day workflow.

For graph neural network, work was done in collaboration with Samuel Hsia. For graph neural networks, we achieve the following:

1. We identify, define, and compute a set of method-agnostic metrics critical to real-world deployment objectives.
2. We propose an evaluation framework to characterize explanation quality along the unique decision boundaries from different GNN models and graph datasets.
3. Based on our empirical analysis, we identify method weaknesses and propose general innovations for improvement.

Finally, for recommendation models, we accomplish the following:

1. We first develop novel methods to explain latent dimensions or embedding layers that are present in virtually all deep learning methods. We create post hoc methods that are easily deployable and understandable to practitioners.
2. We adapt current post hoc methods such as LIME, SHAP, and Vanilla Gradients to work specifically in neural recommendation contexts, switching the input to be embedding layers. Combining this with the previous step, we create multiple possible post hoc explanation methods for neural recommendation models.
3. We generate offline metrics and test our combination of methods among these offline methods for two most commonly used neural recommender architectures: neural collaborative filtering and wide deep.

4. We conduct a user survey to test the online efficacy of our methods, achieving over $N = 50$ diverse samples.

5. Lastly, we combine the results of our offline and online experimentation to discuss which methods to use in which contexts. Thus, we take important steps in resolving the disagreement problem within explainable recommender systems.

1

Related Work

1.1 Explainability and interpretability

It is important for our purposes to create a distinction between explainability and interpretability in machine learning applications. We treat intereptability as a method towards achieving explainability. Explainabile AI refers or XAI refers to models in which resutls can be understood my normal humans. However, this does not mean that the model needs

to be inherently interpretable. That is, there are two classes: models that are interpretable and non-interpretable. Interpretable models include linear regression, logistic regression, decision trees, and set of rules. Practitioners are able to easily discern how the model is outputting either its label or value. Then, there are series of non-interpretable models including random forrests, feed forward neural networks, and deep learning architectures used for image processing and natural languge processing. For these, we utilize post hoc explanations in which we attempt to understand the path through which the complex model used to make the decision after the prediction has been made. This often through analyzing gradients of the model or developing surrogate models to approximate behavior in local regions. More of this discussed in the next subsection.

In general, the distinction between interpretability and explainability follows human thought processes as well. For instance, sometimes, humans utilize careful, logical reasoning before coming to a conclusion. Thus, the reasoning can be inspected step-by-step. Other times, people make intuition-based decisions first on the basis of complex data from long timespans. The humans then seek an explanation for a decision similar to how we utilize post hoc explanations on black box models. Understanding which approach is better for explainable AI likely requires significant breakthroughs in human cognitive science and our understanding about how the human brain works.

1.2 POST HOC METHODS TO ACHIEVE EXPLAINABILITY

We want to explain the predictions of a machine learning model. To achieve this, we need to conceptualize what it means to generate an explanation. An explanation usually relates the feature values of an instance to its model prediction in a humanly understandable way, such as stating that certain features are more important than each other or stating how an input feature can positively or negatively impact the magnitude of the output. Explanations do not only need to relate to feature importance. For instance, they can also be specific instances or sets of instances that are influential. They can be sets of rules or natural English languages. Indeed, it is the diversity of types of explanations that creates challenges in homogeneous evaluation. Ultimately, throughout this thesis, we utilize the feature importance definition primarily.

Given that we would like to focus on complex recommendation modeling, specifically neural-based approaches, we focus our thesis work on post hoc methods. Post hoc methods can be categorized as model-agnostic or model-dependent and global or local.

Model-dependent interpretation tools are specific to a single model or group of models. In contrast, model-agnostic tools can be used on any machine learning model no matter how complicated. These agnostic methods usually work by analyzing feature input and output pairs. Model-agnostic methods often benefit from the fact that some practitioners may not have access to the specific weights and parameters of the original model.

Global methods describe the average behavior of a machine learning model. Global methods are often expressed as expected values based on the distribution of the data. Local interpretation methods explain individual predictions. Local methods can often be roughly aggregated into global methods. In the following paragraphs, we discuss certain archetype examples of each type of method.

1.2.1 Global Methods:

Below we list two examples of global methods. However, this thesis will **focus primarily on local methods** given they are mode wide-spread and can often be units to eventually be used to create a global method.

Partial Dependence Plot:

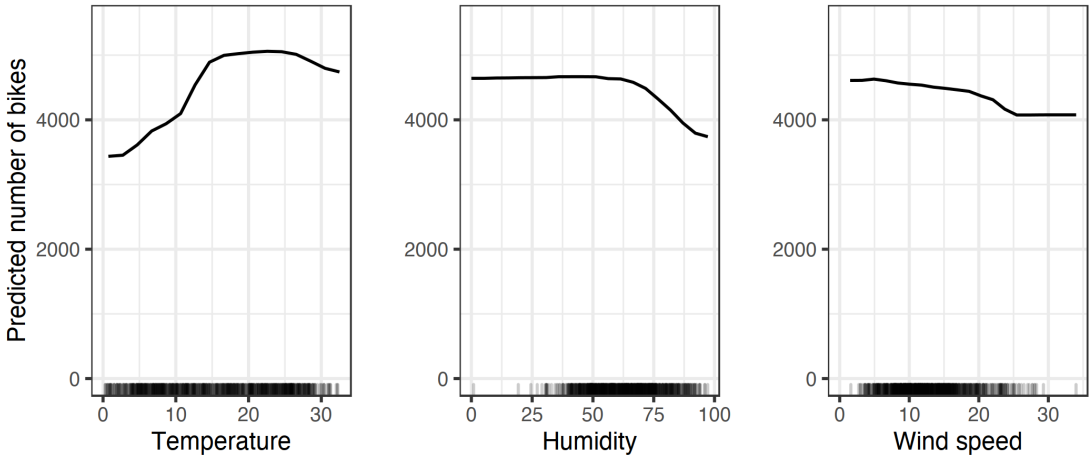

Figure 1.1: Example of partial dependence plots showing how three features (temperature, humidity, wind speed) have a relationship with the target of number of bikes

A partial dependence plot (PDP) shows the marginal effect one or a few features have on

the outcome predicted by a model.

The partial dependence function for regression is defined as:

$$\hat{f}_S(x_S) = E_{X_C}\left[\hat{f}(x_S, X_C)\right] = \int \hat{f}(x_S, X_C)\, d\mathbb{P}(X_C)$$

The x_S are the features for which the partial dependence function should be plotted, and x_C are the other features used in the machine learning model $\hat{f}$, which are here treated as random variables. The feature(s) in S are those for which we want to know the effect on the prediction. The feature vectors x_S and x_C combined make up the total feature space x. This method makes use of the statistical method of marginalization in which we marginalize only over one or two features in S instead of the entire distribution of features in C.

The partial function $\hat{f}_S$ is estimated by calculating averages in the training data, also known as the Monte Carlo method[15]:

$$\hat{f}_S(x_S) = \frac{1}{n}\sum_{i=1}^{n}\hat{f}\left(x_S, x_C^{(i)}\right)$$

.

Global Surrogate Model:

A global surrogate model is an interpretable model that is trained to approximate the predictions of a black box model. We can draw conclusions about the black box model by

interpreting a simpler, inherently explainable model.

Perform the following steps to obtain a surrogate model[48]:

1. We either take the original training dataset for our complex or black box model or generate a new dataset. The new dataset can be generated as a subset or as a data with added noise as long as it is from the same distribution
2. Generate new or use existing predictions
3. Select an interpretable model, often either a linear regression model or a decision tree
4. Train the model and interpret coefficients or leaves

Local Methods:

LIME:

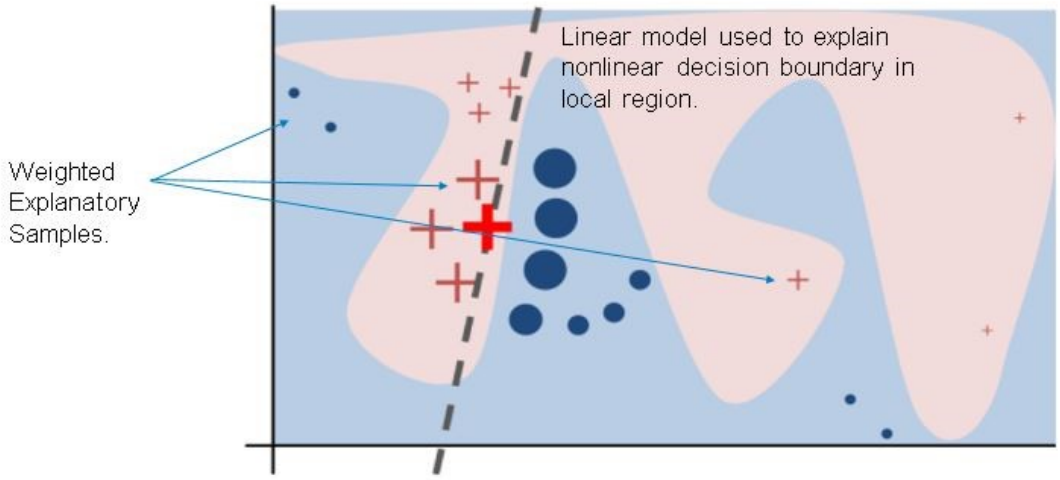

Figure 1.2: Visual representation of LIME creating a local linear approximation of a non-linear, complex decision boundary

LIME or local interpretable model-agnostic explanations is one of the most well-known post hoc methods. The idea of LIME is to create a surrogate model in a local region near

a specific target point to explain the importance of the input features. This is often more accurate than a global surrogate model, as complex black boxes tend to exhibit more linear or simple behavior in local neighborhoods. For LIME, we create new samples by adding normally distributed noise to a target sample, and thereby creating a local neighborhood. We then follow a similar process as with the global method, opting to only use a linear model. [40].

KernelSHAP:

Shapley values rest on the idea that a prediction can be explained by assuming that each feature value of the instance is a 'player' in a game where the prediction is the payout. Shapley values – a method from coalitional game theory – tells us how to fairly distribute the "payout" among the features.

A player can be an individual feature value such as demographics for tabular data. A player can also be a group of feature values. For example to explain an image, pixels can be grouped to super pixels and the prediction distributed among them, much to mirror the way convolution acts on groups of features. One innovation that SHAP brings to the table is that the Shapley value explanation is represented as an additive feature attribution method, a linear model. SHAP specifies the explanation as:

$$g(z') = \varphi_0 + \sum_{j=1}^{M} \varphi_j z'_j$$

where g is the explanation model, $z' \in \{0,1\}^M$ is the coalition vector, M is the maximum coalition size, and $\varphi_j \in \mathbb{R}$ is the feature attribution for a feature j, the Shapley values. A coalition vector simply refers to simplified or chosen features. In the coalition vector, an entry of 1 means that the corresponding feature value is "present" and 0 that it is "absent". To compute Shapley values, we simulate that only some features values are present and some are not. The representation as a linear model of coalitions is a trick for the computation of the φ's. For x, the instance of interest, the coalition vector x' is a vector of all 1's, i.e., all feature values are "present". The formula simplifies to:

$$g(x') = \varphi_0 + \sum^{M} \varphi_j$$

KernelSHAP estimates for an instance x the contributions of each feature value to the prediction. KernelSHAP's estimation is the result of the fact that it is often computationally infeasible to fully calculate SHAP values, especially given the exponential relationship the complexity has with the number of input features. KernelSHAP consists of 5 steps[30]:

1. Sample coalitions $z'_k \in \{0,1\}^M, \quad k \in \{1,\ldots,K\}$ (1 = feature present in coalition, 0 = feature absent).
2. Get prediction for each z'_k by first converting z'_k to the original feature space and then applying model $\hat{f}: \hat{f}(h_x(z'_k))$
3. Compute the weight for each z'_k with the SHAP kernel.
4. Fit weighted linear model.

5. Return Shapley values φ_k, the coefficients from the linear model

Vanilla Gradients:

The Vanilla Gradient method was first created as a pixel attribution method. Often utilized to explain convolutional models, the method can extend to models on tabular data. We calculate the gradient of the loss function for the class we are interested in with respect to the input values, be those of pixels or other data. This gives us an array of the sizes of the input features with negative to positive values. The steps are as follows[35]:

1. Perform a forward pass of the target input
2. Compute the gradient of class score of interest with respect to the input:

$$E_{grad}(I_0) = \left.\frac{\delta S_c}{\delta I}\right|_{I=I_0}$$

   Here we set all other classes to zero.

3. Visualize the gradients. You can either show the absolute values or highlight negative and positive contributions separately.

Mathematically, given some input I and a complex black model that gives it a score $S_c(I)$ for class c, the idea behind using the gradient is that we can approximate this score by applying a first order Taylor expansion

$$S_c(I) \approx w^T I + b$$

where w is the derivative of our score:

$$w = \left. \frac{\partial S_C}{\partial I} \right|_{I_0}$$

1.3 Evaluation of explainability

In general, there are two main types of evaluation methods for explainability[10]:

1. Online experiments. Online experiments can be application level or human level. Application level explanations are put into the product or use-case and tested by a real end user. Human-level tend to be surveys or simplified applications used by any layman.
2. Offline metrics or function-level evaluation (proxy task) does not require humans. This works best when the class of model used has already been evaluated by someone else in a human level evaluation. In creating a function level evaluation, we next consider preferable desidrata of explanation methods and then individual explanations.

For formalizing the disagreement problem, we conduct both online and offline experiments. Similarly, we do offline and online experiments for analyzing post hoc explainability in recommendation models. For graph neural networks, we only conduct offline experiments. This is because it is not intrinsically clear that there is one application or even a small representative set of applications that would lend itself to the entire model class.

Next, we move onto offline properties of individual explanations that are desirable. It is important to note, these are in addition to desirable desiderata of normal machine learning models such as accuracy[42].

1. Fidelity: How well does the explanation approximate the prediction of the black box model? Fidelity is often measured by masking the explanation's outputed most important features and checking if the output of the mdoel has altered.
2. Consistency: How much does an explanation differ between models that have been trained on the same task and that produce similar predictions? One may check between models that have similar hyperparameters or similar model classes.
3. Stability: How similar are the explanations for similar instances? While consistency compares explanations between models, stability compares explanations between similar instances for a fixed model.

We test each of these for both the resolution of the graph neural network disagreement and the recommendation model disagreement. However, we also add two additional metrics: scalability and fairness (fairness only for graph neural networks). This is because we think, especially as we consider explainability methods being deployed in real-world applications, these are key desiderata that must be considered.

With regard to online experiments, many user studies assess how well humans can understand and utilize explanations [10]. Kaur et al. [25] show that data scientists do not have a good understanding of the state-of-the-art interpretability techniques, and are unable to effectively leverage them in debugging ML models. Bhatt et al. [4], conduct a survey to understand the use-cases for local explanations.

Hong et al. [21] conduct a similar survey to identify a variety of stakeholders across the model lifecycle, and higlight core goals: improving the model, making decisions, and building trust in the model. Furthermore, Lakkaraju & Bastani [28] study if misleading explana-

tions can fool domain experts into deploying racially biased models. Similarly, Poursabzi-Sangdeh et al. [39] find that supposedly-interpretable models can lead to a decreased ability to detect and correct model mistakes. Lage et al. [27] use insights from rigorous human-subject experiments to inform regularizers used in explanation algorithms. However, none of these works focus on understanding if and how often practitioners face explanation disagreement and how they resolve it.

1.4 Explainability in GNNs

A wide range of post hoc explanation methods have been proposed for GNNs. These methods specifically look to understand the importance of input nodes, edges, and node features within graphs. At a high level, we can group these methods into four main families: gradient/feature-based, perturbation based, decomposition methods, and surrogate methods[63].

Gradient/feature-based methods such as Guided-BP[62] and CAM [37] use gradients and hidden layer transformed features as proxies for input importance. Perturbation methods such as GNNexplainer[61] and PGexplainer[31] study output variations with respect to input perturbations. Decomposition methods such as Excitation BP[37] , LRP[2], and GNN-LRP[47] measure feature importance by taking predictions and splitting them into several terms. Surrogate methods such as GraphLIME[23] and PGM-explainer[56] employ interpretable,

small models to approximate complex predictions.

In order to evaluate method performance, a few metrics such as accuracy, fidelity/faithfulness, and stability have been proposed[46,63,38]. However, these existing metrics lack a consistent mathematical definition generalizable to all methods and fail to address real-world deployment challenges such as bias, lack of ground truth explanations, and computation time. Furthermore, little work has been done to empirically evaluate these metrics across various datasets and GNN models. To our knowledge, only one recent work[46] evaluated the performance of gradient/feature-based methods along this GNN model and dataset framework. However, this work only considers one family of explainability methods and does not address the limitations of attribution (e.g. inability to jointly consider graph structure and node feature information) with respect to other methods of explainability.

1.5 Recommendation models:

A recommender system is a subclass of information filtering system that seeks to predict the rating or preference a user would give to an item. In any recommender system, the most basic unit is a user-item interaction.

The earliest recommender systems used content-based filtering methods or memory-based collaborative filtering methods[14]. In user-based collaborative filtering, we postulate that users with similar historical ratings should also have similar interests. Thus, if a cer-

tain user needs to decide whether to consume a new item, we aggregate information from those similar users. There are three steps. First, we calculate similar users which can be done through euclidean distance, cosine similarity, or some other metric of rating vectors. Second, we find the K nearest neighbors based on that distance. Third, we aggregate ratings of the neighbors to predict rating of the item.

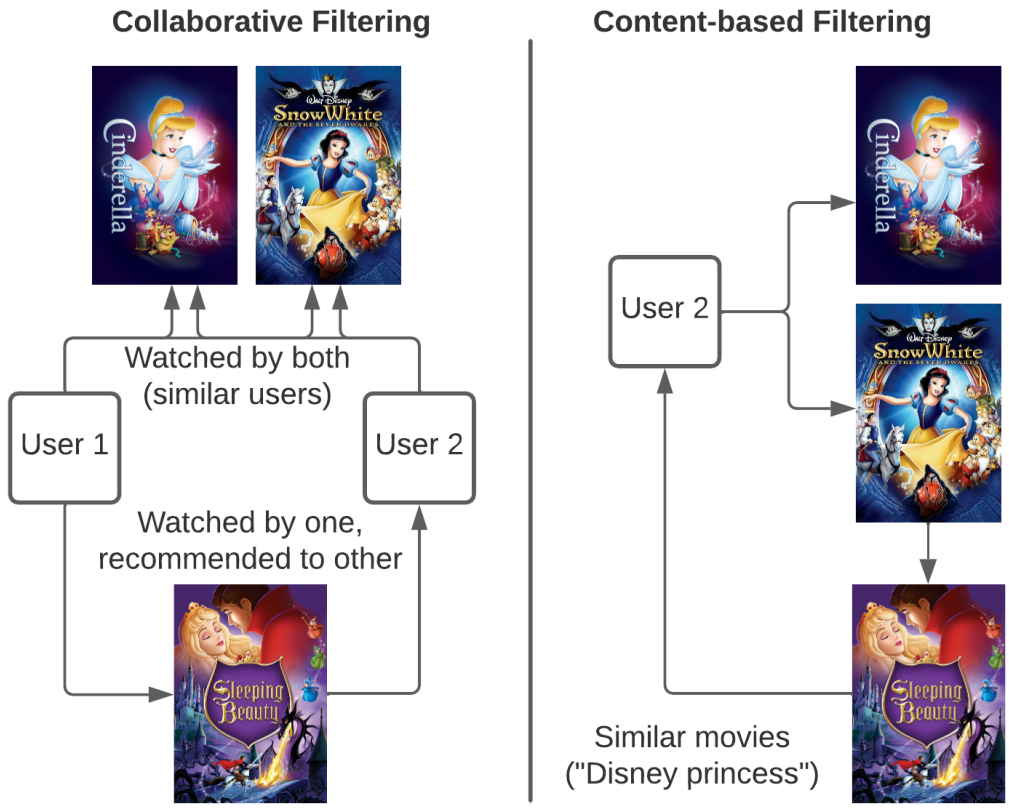

Figure 1.3: Differences between collaborative and content-based recommendation models

As machine learning became more widespread, model-based approaches to collaborative

filtering appeared[64]. Specifically, memory based content filtering is not suitable for practical applications when dealing with large amounts of users and items. Although there are a variety of model methods for approaching collaborative filtering, the most commonly used was matrix factorization. In matrix factorization, users and items are projected into a latent space. Each user and item are represented by a latent vector. Then, to understand how a user would rate a certain item, matrix factorization proposes utilizing the inner product of the latent vectors.

With the advent of larger and larger data sources for recommendation systems, especially in e-Commerce or social media platforms, practitioners have moved past simple collaborative filtering models towards neural-based methods. Neural based methods are often either non-temporal or temporal.

For non-temporal methods, neural based methods tend to have two components: representation learning and interaction modeling[64]. Beginning with representation learning, the objective is to learn a user embedding matrix $\mathbf{P}$ and an item embedding $\mathbf{Q}$, with $\mathbf{p}_u$ and $\mathbf{q}_i$ denote the representation parameters for user u and item i, respectively. Differences in the representation approach depend on both the type of input data and the modeling technique. Generally, there are three primary representation learning techniques: history behavior aggregation enhanced models, autoencoder-based models, and graph learning approaches.

1.6 Explainability in recommendation models

Existing explainable recommendation research is divided into the following two types: one is the way to display the explanation, and the other is a model that generates explainable recommendations. There are a variety of explainable styles[67]. Prototypical examples include text explanations with reasons why a recommendation was given (e.g, "because you are a 17 year old..."), word clouds, or potentially listing examples instances or users that were influential in the recommendation.

In terms of methodologies to explain recommendations, we touch on factorization models, topic models, graph models, deep learning based approaches, and finally post hoc explainability methods.

In matrix factorization explainability methods, the aformentioned matrix factorization algorithms are extended to explain the latent representations of users and items. In the earliest iteration of explainability for matrix factorization, Zhang et al. proposed Explicit Factor Models (EFM)[68]. The idea here is to discover a user's favorite item features and then utilize those as explanations. Specifically, the proposed approach extracts explicit product features from user reviews, and then aligns each latent dimension of matrix factorization with an explicit feature. Then, the explanation becomes: "we think you would enjoy this product because you are interested in X feature." Chen et al. (2016) further extended the EFM model to tensor factorization and utilized textual reviews to construct a user-item feature

cube[5]. Learn to rank was then utilized on the cube to pairwise optimize user preferences on features and items. Further extensions included the use of multi-task learning and attention techniques to tune the user attention distribution over features on different items[67].

A Topic Model (TM) is a statistical model which aims to discover and annotate large text corpora with thematic information. The first kind of topic model was the Latent Dirichlet Allocation technique (LDA)[24]. The basic idea is that every document is about several topics and that each word in the document can be associated with one of these topics. Every topic is seen as a probability distribution over the corpus dictionary, and every document can be represented with a probability distribution over. Similarly a recommendation can be seen as a document of past history.

Because many user-item relationships can be represented as graphs, many explainability methods in recommendation model specifically focus on graphical methods. He et al. (2015) introduced a tripartite graph structure to model the user-item-aspect ternary relations for top-N recommendation[19]. In TriRank, explanations are attributed to the top ranked aspects that match the target user and the recommended item. Heckel et al. (2017) conducted over-lapping co-clustering based on user-item bipartite graph. Unlike the previous algorithm, this does not require textual reviews[19].

The least explored but potentially most promising method is deep learning methods for recommendation explainability. Seo et al. (2017) proposed to model user preferences and

item properties using convolutional neural networks[49]. When predicting the user-item rating, the model selectively chooses review words with different attention weights when then allows the explanation to focus on the words that are the most important. Wu et al. (2019) extended this to provide content summaries as opposed to just key words[60]. Rather than highlighting review words as explanations, Costa et al. (2018) proposed a method for automatically linguistic explanations based on character-level RNN which has later been extended to include visual data as well[8].

Finally, we focus on post hoc methods. Post hoc methods have become increasingly more important given the widespread use of deep learning recommendation models. Methods described above which often focus on aligning latent dimensions of matrix factorization models with review data can often not be applied when those latent dimensions go through a series of fully connected layers. Peake and Wang (2018) provided an association rule mining approach. The authors treated an arbitrary recommendation model as the center of their method to make it model-agnostic[36]. Singh and Anand (2018) used a surrogate method[52]. Technically, the authors first train a black box ranker and then use the ranking labels produced by the ranker as secondary training data to train an explainable tree-based model. The tree-based model is the post hoc explanation model to generate explanations for the ranking list. Wang et al. (2018) proposed a model-agnostic reinforcement learning method[58].

2

The Disagreement Problem: Offline Online Experiments

2.1 Understanding and measuring disagreement between model explanations

In this section, we discuss practitioner perspectives on what constitutes disagreement between two explanations, and then formalize the notion of explanation disagreement. To

this end, we first describe the study that we carry out with data scientists to understand what constitutes explanation disagreement, and the extent to which they encounter this problem in practice. We then discuss the insights from this study, and leverage these insights to propose a novel framework which can quantitatively measure the disagreement between any two explanations.

2.1.1 Characterizing Explanation Disagreement Using Practitioner Inputs

Interviews with Practitioners: Study Details

We conducted 30-minute long semi-structured interviews with 25 data scientists who employ explainability techniques to understand model behavior and explain it to their customers and managers. All of these data scientists were employed in for-profit organizations, and worked for various companies in the technology and financial services sectors in the United States. Furthermore, all the participants used state-of-the-art (local) post hoc explanation methods such as LIME, SHAP, and gradient based methods in their day-to-day workflow. 19 of these participants (76%) were male, and 6 of them (24%) were female. 16 participants (64%) had more than 2 years of experience working with explainability techniques, and the remaining 9 (36%) had about 8 to 12 months of experience. Our interviews included the following major research questions:

1. RQ 1: How often do you use multiple explanation methods to understand the same model prediction?

2. RQ 2: What constitutes disagreement between two explanations that explain the same model prediction?
3. RQ 3: How often do you encounter disagreements between explanations output by different methods for the same model prediction?

2.1.2 Findings and Insights

Our study revealed a wealth of information about how data scientists utilize explanation methods and their perspectives on disagreement between explanations. 22 out of the 25 participants (88%) said that they almost always use multiple explanation methods to understand the same model prediction. Furthermore, 21 out of the 25 participants (84%) mentioned that they have often run into some form of disagreement between explanations output by different methods for the same prediction. They also elaborated on when they think two explanations disagree:

Top features are different: Most of the popular post hoc explanation methods (e.g., LIME, SHAP, Gradient based methods) return a feature importance value associated with each feature. These values indicate which features contribute the most either positively or negatively (i.e., the top features) to the prediction. 21 out of the 25 participants (84%) in our study mentioned that such a set of top features is *"the most critical piece of information"* that they rely on in their day-to-day workflow. They also noted that they typically look at

the top 5 to 10 features provided by an explanation for each prediction. When two explanations have different sets of top features, they consider it to be a disagreement.

Ordering among top features is different: 18 out of 25 participants (72%) in our study indicated that they also consider the ordering among the top features very carefully in their workflow. Therefore, they consider a mismatch in the ordering of the top features provided by two different explanations to be a disagreement.

Direction of top feature contributions is different: 19 out of 25 participants (76%) mentioned that the *sign* or *direction* of the feature contribution (is the feature contributing positively or negatively to the predicted class?) is another critical piece of information. Any mismatch in the signs of the top features between two explanations is a sign of disagreement. As remarked by one of the participants, "*I saw an explanation indicating that a top feature bankruptcy contributes positively to a particular loan denial, and another explanation saying that it contributes negatively. That is a clear disagreement. The model prediction can be trusted with the former explanation, but not with the latter.*".

Relative ordering of certain features is different: 16 of our study participants (64%) indicated that they also look at relative ordering between certain features of interest; and if explanations provide contradicting information about this aspect, then it is considered a disagreement. For example, one of the participants remarked, *"I often check if salary is more important than credit score in loan approvals. If one explanation says salary is more*

important than credit score, and another says credit score is more important than salary; then it is a disagreement."

A very striking finding from our study is that participants typically characterize explanation disagreement based on factors such as mismatch in top features, feature ordering, and directions of feature contributions, but not on the feature importance values output by different explanation methods. 24 out of 25 participants (96%) in our study opine that feature importance values output by different explanation methods are not directly comparable. They also note that this is due to the fact that while LIME outputs coefficients of a linear model as feature importance values, SHAP outputs Shapley values as feature attributions which sum to the probability of the predicted class. So, they don't try to base explanation disagreement on these numbers not being equal or similar. One of our participants succinctly summarized practitioners' perspective on this explanation disagreement problem – *"The values generated by different explanation methods are clearly different. So, I would not characterize disagreement based on that. But, I would at least want the explanations they output to give me consistent insights. The explanations should agree on what are the most important features, the ordering among them and so on for me to derive consistent insights. But, they don't!"*

2.1.3 Formalizing the Notion of Explanation Disagreement

Our study indicates that ML practitioners consider the following key aspects when they think about explanation disagreement: a) the extent to which explanations differ in the top-*k* features, the signs (or directions of contribution) and the ordering of these top-*k* features, and b) the extent to which explanations differ in the relative ordering of certain features of interest. To capture these intuitions about explanation disagreement, we propose six different metrics, namely, *feature agreement*, *rank agreement*, *sign agreement*, *signed rank agreement*, *rank correlation*, and *pairwise rank agreement*. While the first four metrics capture disagreement w.r.t. the top-*k* features of the explanations, the last two metrics capture disagreement w.r.t. a selected set of features which could be provided as input by an end user.

Measuring Disagreement w.r.t. Top-k Features

We now define four metrics, which capture specific aspects of explanation disagreement w.r.t. the top-*k* features.* For all metrics, lower values indicate higher disagreement.

Feature Agreement: ML practitioners in our study clearly indicated that a key notion of disagreement between a pair of explanations is that they output different top-*k* features. To capture this notion, we introduce the feature agreement metric which computes the

*The top-*k* features of an explanation are typically computed only based on the magnitude of the feature importance values and not the signs.

fraction of common features between the sets of top-k features of two explanations. Given two explanations E_a and E_b, the feature agreement metric can be formally defined as:

$$\mathit{FeatureAgreement}(E_a, E_b, k) = \frac{|\mathit{top\_features}(E_a, k) \cap \mathit{top\_features}(E_b, k)|}{k}$$

where $\mathit{top\_features}(E, k)$ returns the set of top-k features (based on the magnitude of the feature importance values) of the explanation E. If the sets of top-k features of explanations E_a and E_b match, then $\mathit{FeatureAgreement}(E_a, E_b, k) = 1$.

Rank Agreement: Practitioners in our study also indicated that if the ordering of the top-k features is different for two explanations (even if the feature sets are the same), then they consider it to be a disagreement. To capture this notion, we introduce the rank agreement metric which computes the fraction of features that are not only common between the sets of top-k features of two explanations, but also have the same position in the respective rank orders. Rank agreement is a stricter metric than feature agreement since it also considers the ordering of the top-k features. Given two explanations E_a and E_b, the rank agreement metric can be formally defined as:

$$\frac{|\bigcup_{s \in S} \{s \mid s \in \text{top features } (E_a, k) \wedge s \in \text{top features } (E_b, k) \wedge \text{rank}(E_a, s) = \text{rank}(E_b, s)\}|}{k}$$

where S is the complete set of features in the data, $\mathit{top\_features}(E, k)$ is defined as above,

and $rank(E, s)$ returns the position (or the rank) of the feature s according to the explanation E. If the rank-ordered lists of top-k features of explanations E_a and E_b match, then $RankAgreement(E_a, E_b, k) = 1$.

Sign Agreement: In our study, practitioners also mentioned that they consider two explanations to disagree if the feature attribution signs or the directions of feature contribution (does a feature contribute positively or negatively to the prediction?) do not align for the top-k features. To capture this notion, we introduce the sign agreement metric which computes the fraction of features that are not only common between the sets of top-k features of two explanations, but also share the same sign (direction of contribution) in both explanations. Sign agreement is a stricter metric than feature agreement since it also considers signs (directions of contributions) of the top-k features. More formally:

$$\frac{\left|\bigcup_{s \in S} \{s \mid s \in \text{top } \textit{features}\,(E_a, k) \wedge s \in \text{top features}\,(E_b, k) \wedge \text{sign}\,(E_a, s) = \text{sign}\,(E_b, s)\}\right|}{k}$$

where $sign(E, s)$ returns the sign (direction of contribution) of the feature s according to the explanation E.

Signed Rank Agreement: This metric fuses together all the above notions, and computes the fraction of features that are not only common between the sets of top-k features of two explanations, but also share the same feature attribution sign (direction of contribution) and position (rank) in both explanations. Signed rank agreement is the strictest compared

to all the aforementioned metrics since it considers both the ordering and the signs (directions of contributions) of the top-k features. More formally, $SignedRankAgreement(E_a, E_b, k)$ can be written as:

$$\frac{|\bigcup_{s \in S}\{s \mid s \in top\_features(E_a, k) \wedge s \in top\_features(E_b, k) \wedge sign(E_a, s) = sign(E_b, s) \wedge rank(E_a, s) = rank(E_b, s)\}|}{k}$$

where *top_features*, *sign*, *rank* are all as defined above. $SignedRankAgreement(E_a, E_b, k) = 1$ if the top-k features of two explanations match on all aspects (i.e., features, feature attribution signs, rank ordering) barring the exact feature importance values.

Measuring Disagreement w.r.t. Features of Interest

Practitioners also indicated that they consider two explanations to be different if the relative ordering of features of interest (e.g., salary and credit score) differ between the two explanations. To formalize this notion, we introduce the two metrics below.

Rank Correlation: We adopt a standard rank correlation metric (i.e., Spearman's rank correlation coefficient) to measure the agreement between feature rankings provided by two explanations for a selected set of features. In practice, this selected set of features corresponds to features that are of interest to end users, and can be provided as input by end

users. Given two explanations E_a and E_b, rank correlation can be computed as:

$$RankCorrelation(E_a, E_b, F) = r_s(Ranking(E_a, F), Ranking(E_b, F))$$

where F is a selected set of features potentially input by an end user, r_s computes Spearman's rank correlation coefficient, and $Ranking(E, F)$ assigns ranks to features in F based on explanation E. Lower values indicate higher disagreement.

Pairwise Rank Agreement: Pairwise rank agreement takes as input a set of features that are of interest to the user, and captures if the relative ordering of every pair of features in that set is the same for both the explanations i.e., if feature A is more important than B according to one explanation, then the same should be true for the other explanation. More specifically, this metric computes the fraction of feature pairs for which the relative ordering is the same between two explanations. More formally:

$$PairwiseRankAgreement(E_a, E_b, F) = \frac{\sum_{i,j \text{ for } i<j} 1[RelativeRanking(E_a, f_i, f_j) = RelativeRanking(E_b, f_i, f_j)]}{\binom{|F|}{2}}$$

where $F = \{f_1, f_2 \cdots\}$ is a selected set of features input by an end user, $RelativeRanking(E, f_i, f_j)$ is an indicator function which returns 1 if feature f_i is more important than feature f_j according to explanation E, and 0 otherwise.

The feature agreement metric captures what fraction of features are common between

the sets of top-k features corresponding to two explanations. For example, Feature agreement measures the concordance of two sets of top-k features computed using two explanations based on shared features. It calculates the proportion of shared features between the two sets of top-k features, out of k features. Specifically,

$$\mathit{FeatureAgreement}(E_a, E_b, k) = \frac{|\mathit{TopFeatures}(E_a, k) \cap \mathit{TopFeatures}(E_b, k)|}{k},$$

where *TopFeatures* computes the top-k features using explanation E_a or E_b. By definition, when $k = |F|$,

FeatureAgreement(E_a, E_b, k) = 1.

2.2 Empirical analysis of explanation disagreement

We leverage the metrics outlined in Section 3 and carry out a comprehensive empirical analysis with six state-of-the-art explanation methods and four real-world datasets to study the explanation disagreement problem. In this section, we describe the datasets that we use (Section 2.2.1), our experimental setup (Section 2.2.2), and key findings (Section 2.2.3).

2.2.1 Datasets

To carry out our empirical analysis, we leverage four well known datasets spanning three different data modalities (tabular, text, and images). For **tabular** data, we use the Correc-

tional Offender Management Profiling for Alternative Sanctions (COMPAS) dataset[3] and the German Credit dataset[11]. This dataset comprises of seven features capturing information about the demographics, criminal history, and prison time of 4,937 defendants. Each defendant in the data is labeled either as high-risk or low-risk for recidivism based on the COMPAS algorithm's risk score. The German Credit dataset contains twenty features capturing the demographics, credit history, bank account balance, loan information, and employment information of 1,000 loan applicants. The class label here is a loan applicant's credit risk (high or low). For **text** data, we use Antonio Gulli (AG)'s corpus of news articles (AG_News)[66]. The dataset contains 127,600 sentences (collected from 1,000,000+ articles from 2,000+ sources with a vocabulary size of 95,000+ words). The class label is the topic of the article from which a sentence was obtained (World, Sports, Business, or Science/Technology). For **image** data, we use the ImageNet−1k[43,44] object recognition dataset. It contains 1,381,167 images belonging to 1000 object categories. We experiment with images from PASCAL VOC 2012[12] which provides segmentation maps that can be directly used as super-pixels for the explanation methods.

2.2.2 Experimental Setup

We train a variety of black box models on the data. In case of tabular data, we train four models: logistic regression, densely-connected feed-forward neural network, random forest, and gradient boosted trees. In case of text data, we train a widely-used vanilla LSTM-based

text classifier on AG_News[65] corpus. For image data, we use the pre-trained ResNet-18 [18] for ImageNet.

Next, we apply six state-of-the-art post hoc explanation methods to explain the black box models' predictions for a set of test data points. We apply two perturbation-based explanation methods ([41] and [30]), and four gradient-based explanation methods ([51], [50], [55], and [53]). In case of explanation methods with a sample size hyper-parameter, we either run the explanation method to convergence (i.e., select a sample size such that an increase in the number of samples does not significantly change the explanations) or use a sample size that is much higher than the sample size recommended by previous work.

We then evaluate the (dis)agreement between the explanation methods using the metrics described in Section 2.1.3. For tabular and text data, we apply rank correlation and pairwise rank agreement across all features; and feature agreement, rank agreement, sign agreement, signed rank agreement across top-k features for varying values of k. For image data, metrics that operate on the top-k features are more applicable to super-pixels. Thus, we apply the six disagreement metrics on explanations output by LIME and KernelSHAP (which leverage super pixels), and calculate rank correlation (across all pixels as features) between the explanations output by gradient-based methods.

2.2.3 Results and Insights

We discuss the results of our empirical analysis for each of the three data modalities.

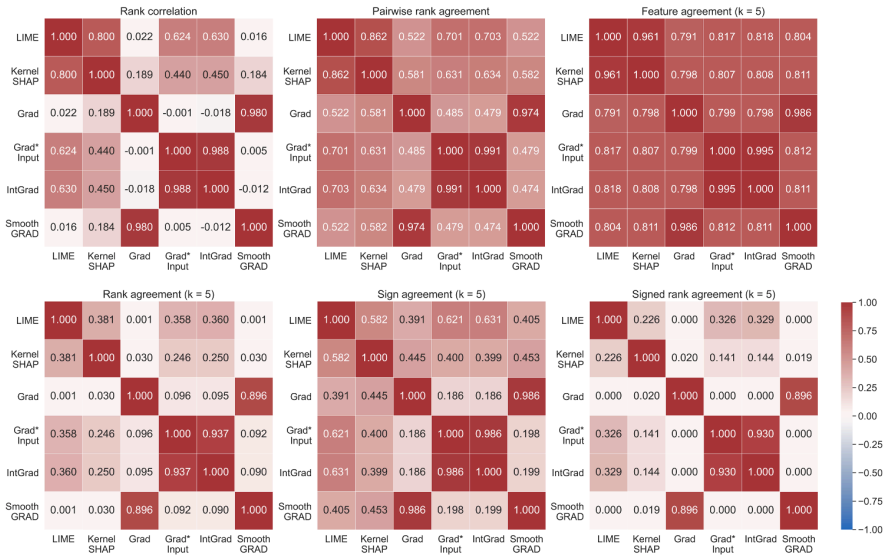

Figure 2.1: Disagreement between explanation methods for neural network model trained on COMPAS dataset measured by six metrics: rank correlation and pairwise rank agreement across all features, and feature, rank, sign, and signed rank agreement across top $k = 5$ features. Heatmaps show the average metric value over test set data points for each pair of explanation methods, with lighter colors indicating stronger disagreement. Across all six heatmaps, the standard error ranges between 0 and 0.009.

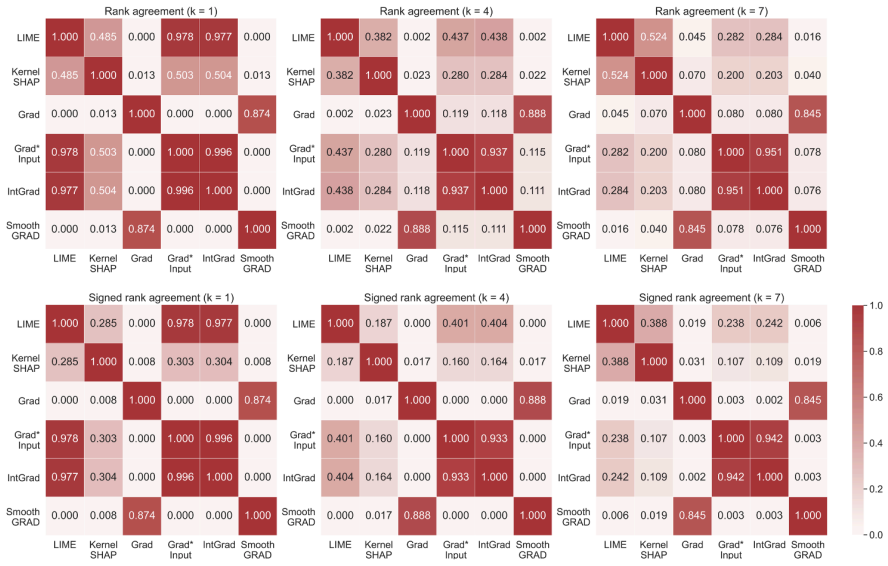

Figure 2.2: Disagreement between explanation methods for neural network model trained on COMPAS dataset measured by rank agreement (top row) and signed rank agreement (bottom row) at top-k features for increasing values of k. Each cell in the heatmap shows the metric value averaged over test set data points for each pair of explanation methods, with lighter colors indicating stronger disagreement. Across all six heatmaps, the standard error ranges between 0 and 0.003.

Tabular Data:

Figure 2.1 shows the disagreement between various pairs of explanation methods for the neural network model trained on COMPAS dataset. We computed the six metrics outlined in Section 2.1.3 where we used $k = 5$ (out of 7 features) for metrics that focus on the top features. Each cell in the heatmap shows the metric value averaged over the test data points for each pair of explanation methods with lighter colors indicating more disagreement. We see that explanation methods tend to exhibit slightly higher values on pairwise rank agreement and feature agreement metrics, and relatively lower values on other metrics (indicating more disagreement).

We next study the effect of the number of top features on the degree of disagreement. Figure 2.2 shows the disagreement of explanation methods for the neural network model trained on COMPAS dataset. We computed rank agreement (top row) and signed rank agreement (bottom row) at top-k features for increasing values of k. We see that as the number of top-k features increases, rank agreement and signed rank agreement decrease. This indicates that, as k increases, top-k features of a pair of explanation methods are less likely to contain shared features with the same rank (as measured by rank agreement) or shared features with the same rank and sign (as measured by signed rank agreement). These patterns are consistent across other models trained on the COMPAS dataset.

In addition, across all metrics, values of k, and models, the specific explanation method

pairs of Grad-SmoothGrad and Grad*Input-IntGrad consistently exhibit strong agreement while the pairs of Grad-IntGrad, Grad-Grad*Input, SmoothGrad-Grad*Input and SmoothGRAD-IntGrad consistently exhibit strong disagreement. This suggests a dichotomy among gradient-based explanation methods, i.e., certain gradient-based explanation methods are consistent with one another while others are inconsistent with one another.

Furthermore, there are varying degrees of disagreement among pairs of explanation methods. For example, for the neural network model trained on the COMPAS dataset, rank correlation displays a wide range of values across explanation method pairs, with 10 out of 15 explanation method pairs even exhibiting negative rank correlation when explaining multiple data points. This is shown in the left panel of Figure 4.15, which displays the rank correlation over all the features among all pairs of explanation methods for neural network model trained on COMPAS dataset.

All the patterns discussed above are also generally reflected in the German Credit dataset. However, explanation methods tend to display stronger disagreement for the German Credit dataset than for the COMPAS dataset. For example, rank agreement and signed rank agreement are lower for the German Credit dataset than for the COMPAS dataset at top 25%, 50%, 75%, and 100% of features for both logistic regression and neural network models. One possible reason is that the German Credit dataset has a larger set of features than the COMPAS dataset, resulting in a larger number of possible ranking and sign com-

binations assigned by a given explanation method and making it less likely for two explanation methods to produce consistent explanations.

Lastly, explanation methods display trends associated with model complexity. For example, the disagreement between explanation methods is similar or stronger for the neural network model than for the logistic regression model across metrics and values of k, for both COMPAS and German Credit datasets. In addition, explanation methods show similar levels of disagreement for the random forest and gradient-boosted tree models. These trends suggest that disagreement among explanation methods may increase with model complexity. As the complexity of the black box model increases, it may be more difficult to accurately approximate the black box model with a simpler model (LIME's strategy, for example) and more difficult to disentangle the contribution of each feature to the model's prediction. Thus, the higher the model complexity, the more difficult it may be for different explanation methods to generate the true explanation and the more likely it may be for different explanation methods to generate differently false explanations, leading to stronger disagreement among explanation methods.

Text Data:

In the case of text data, we deal with a high-dimensional feature space where words are features. We plot the six metrics for $k = 11$ which is 25% of the average text length of a sentence (data point) in the dataset (Figure 2.4). As can be seen, we observe severe disagree-

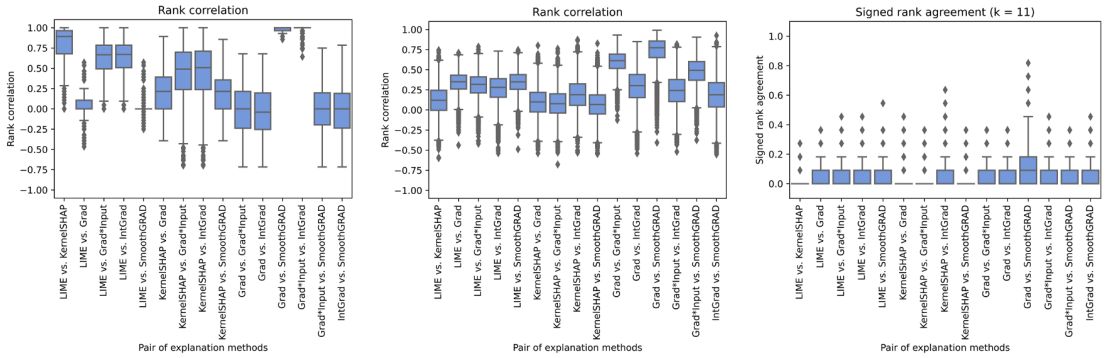

Figure 2.3: Distribution of rank correlation over all features for neural network model trained on COMPAS (left), and rank correlation across all features (middle) and signed rank agreement across top-11 features (right) for neural network model trained on AG_News.

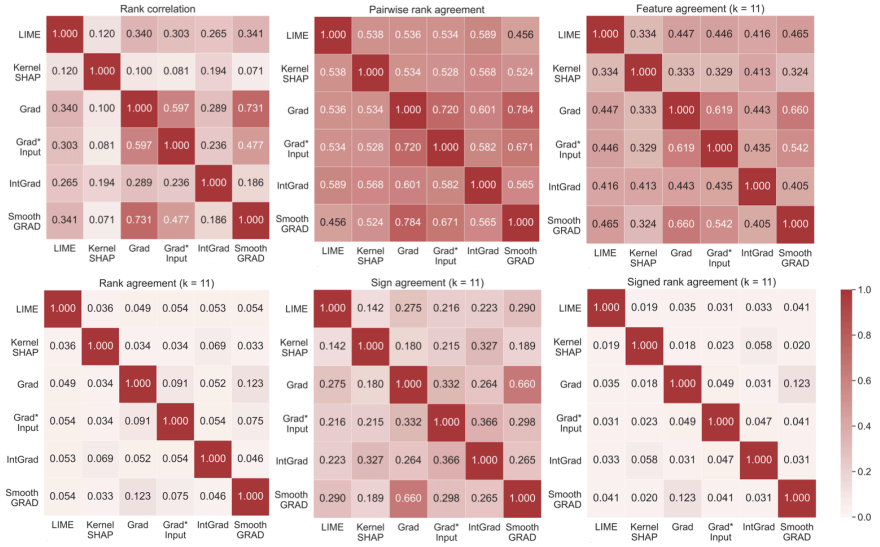

Figure 2.4: Disagreement between explanation methods for the LSTM model trained on the AG_News dataset using $k = 11$ features for metrics operating on top-k features, and all features for other metrics. Each heatmap shows the metric value averaged over test data for each pair of explanation methods. Lighter colors indicate more disagreement. Standard error ranges from 0.0 to 0.0025 for all six metrics.

ments across all the six disagreement metrics. Rank agreement and signed rank agreement are the lowest between explanations with values under 0.1 for most cases indicating disagreement in over 90% of the top-k features. Trends are quite similar for rank correlation

and feature agreement with better agreement between gradient based explanation methods, such as a feature agreement of 0.61 for Grad*Input and Gradient method.

In addition to the overall disagreements, we also notice specific patterns of agreement between a group of explanation methods. Based on middle and right panels of Figure 4.15, we notice that there is a high rank correlation between pairs of gradient based explanations. Although Integrated Gradients has the lowest correlation with the rest of the gradient methods, still this correlation is significantly higher compared to its correlation with KernelSHAP and LIME. We also notice that disagreement is lower between LIME and other explanation methods compared to KernelSHAP and other methods (e.g., rank correlation of 0.4-0.6 for LIME as opposed to 0.2-0.4 for KernelSHAP. Finally, we see a higher disagreement in explanation methods for text compared to tabular data which suggests that disagreement may worsen as the number of features increases. We also observe a similar agreement pattern between LIME and other methods which was also observed earlier in our experiments with tabular datasets, hence, indicating that LIME explanations are most aligned with other post hoc explanation methods.

Image Data:

While LIME and KernelSHAP consider super pixels of images as features, gradient based methods consider pixels as features. Furthermore, the notion of top-*k* features and the metrics we define on top-*k* features are not semantically meaningful when we consider pixels as

features. Given this, we compute all the six metrics to capture disagreement between explanations output by LIME and KernelSHAP with super pixels as features. We also compute rank correlation on all the pixels (features) to capture disagreement between explanations output by gradient based methods.

Unlike earlier trends with tabular and text data, we see higher agreement between KernelSHAP and LIME on all the six metrics: rank correlation of 0.8977, pairwise rank agreement of 0.9302, feature agreement of 0.9535, rank agreement of 0.8478, sign agreement of 0.9218 and signed rank agreement of 0.8193. However, the trends are quite opposite when we compute rank correlation at pixel-level for gradient-based methods. For instance, rank correlation between Integrated Gradients and SmoothGrad is 0.001 (indicating high disagreement). The disagreement is similarly quite high in case of other pairs of gradient based methods. This suggests that disagreement could potentially vary significantly based on the granularity of image representation.

2.3 Resolving the disagreement problem in practice: a qualitative study

In order to understand how practitioners resolve the disagreement problem, we conduct a qualitative user study targeted towards explainability practitioners. We now describe our user study design and discuss our findings.

2.3.1 User Study Design

In total, 25 participants participated in our study, 13 from academia and 12 from industry. Participants from academia were graduate students, and postdoctoral researchers, while participants from industry were data scientists and ML engineers from three different firms. 20 of these participants indicated that they have used explainability methods in their work in a variety of ways, including doing research, helping clients explain their models, and debugging their own models. Following the setup in Section 2.2, we asked participants to compare the output of five pairs of explainability methods on the predictions made by the neural network we trained on the COMPAS dataset. We chose the COMPAS dataset because it only has 7 features, making it easy for participants to understand the explanations.

First, the participants are shown an information page explaining the COMPAS risk score binary prediction setting and various explainability algorithms. We indicate that we trained a neural network to predict the COMPAS risk score (low or high) from the seven COMPAS features. We also give a brief description of each of these seven features to the participant and tell them to assume that the criminal defendant's risk of recidivism is correctly predicted to be high risk. In this information page, we also briefly introduce and summarize the six explainability algorithms we use in the study (LIME, KernelSHAP, Gradient, Gradient*Input, SmoothGrad, and Integrated Gradients). Finally, we provide links to the papers describing each of the algorithms.

Next, each of the participants is shown a series of 5 prompts, a sample of which is shown in Figure 2.5. Each prompt presents two explanations of our neural network model's prediction corresponding to a particular data point generated using two different explanation methods (e.g., LIME and KernelSHAP in Figure 2.5). For each of the 15 pairs of explanation methods, we chose a different data point from COMPAS to run the two methods on, giving us a set of 15 prompts. These prompts were picked to showcase various levels of agreement. We display the full set of $k = 7$ COMPAS features, showing the feature importance of each feature. Red and blue bars indicate that the feature contributes negatively and positively respectively to the predicted class. The participants were first asked the question *"To what extent do you think the two explanations shown above agree or disagree with each other?"* and given four choices: *completely agree, mostly agree, mostly disagree*, and *completely disagree*. If the participant indicated any level of disagreement (any of the latter 3 choices), we then asked *"Since you believe that the above explanations disagree (to some extent), which explanation would you rely on?"* and presented with three choices: the two explainability methods shown and *"it depends"*. They were then asked to explain their response. The users were allowed to take as much time as they wanted to complete the study.

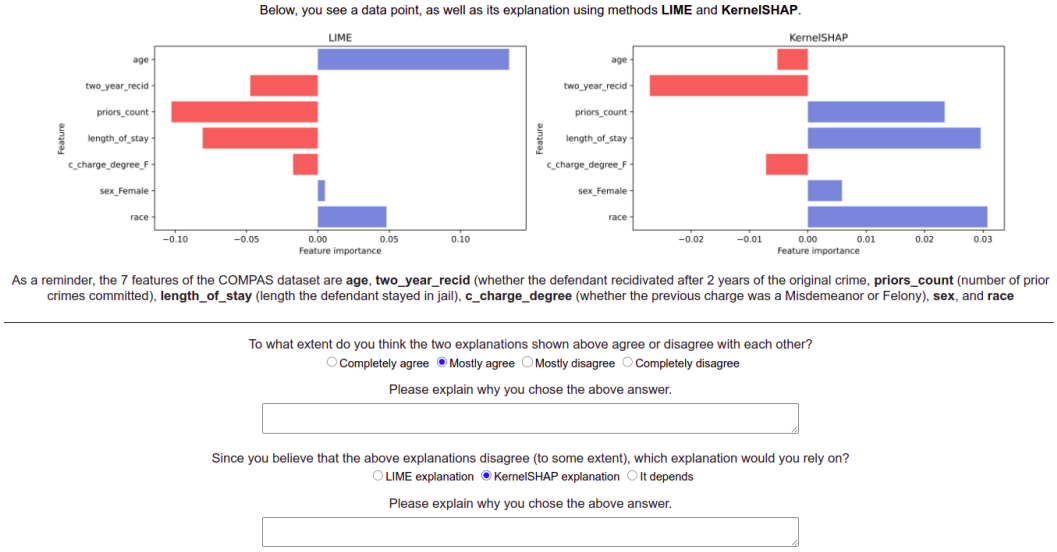

Figure 2.5: The user interface for a prompt. The user is shown two explanations for a COMPAS data point, showing the feature importance value of each of the 7 features. Red and blue indicate negative and positive feature values, respectively. See the text for more details.

3

Graph Neural Networks

3.1 Preliminaries

Here we define important GNN terminology. First, let G denote a graph on edges E and nodes V that are associated with node features $\mathcal{X} = \{x_1, \ldots, x_n\}, x_i \in \mathbb{R}^d$. In this paper, we focus on node classification, which is formulated by approximating the following function: $f : V \mapsto \{1, \ldots, C\}$. Thus, let y_i represent the original class prediction for node i

.

We use a GNN to approximate the function represented by φ. GNNs have three main computation steps that are present in a wide range of model architectures; the ultimate goal is to calculate embedding representations of every node throughout the network. The first step is a message calculation that is computed by a neural network for every pair of node hidden representations. Second, we have an aggregation which essentially pools together all the hidden representations. Finally, there is a neural update function that combines the target node's hidden representation with the aggregated message. The final hidden representations can be used for node classification.

We consider the general problem of an explanation method E creating some feature attribution A that assigns importance scores to each of the d -dimensional node features $\mathcal{X} = \{x_1, \ldots, x_n\}, x_i \in \mathbb{R}^d$. Thus the attribution vector is also d-dimensional. Often, we use attribution vectors to mask features. Specifically, let $m_{i,k}$ represent the feature mask with only the top k important features and $1 - m_{i,k}$ represent the feature mask that removes the top k important features. In the future we can extend this to node attribution that is of size $|V|$ or edge attribution which is of size $|E|$.

3.2 Problem statement methodology

Given our goal is to develop a way to compare, contrast, and chose explainability methods given a wide array of use cases, we decompose our problem statement into three specific questions.

1. How do we quantitatively evaluate explainability methods to cover varied metrics of success?
2. What is the "best" explainability method suited for a particular circumstance?
3. What are general improvements for each method based on a certain metric?

We characterize the quality of three representative explainability methods on different GNN use-cases – more specifically, on unique decision boundaries generated from different GNN models and graph datasets. We propose a set of quantitative metrics that can be used the evaluate explanations along the following properties: *fidelity*, *consistency*, *stability*[*], *fairness* and *scalability*.

3.2.1 Metrics:

Fidelity: The difference between the original predictions and the new predictions after masking out important input features.

As mentioned in Preliminaries, let y_i represent the original prediction for node i from the GNN and $y_i^{1-m_{i,k}}$ represent the new prediction for node i from the GNN with k impor-

[*]Model stability is covered in the Appendix.

tant features masked.

$$Fidelity = \frac{1}{|V|}\sum_{i=1}^{|V|} 1(y_i = y_i^{1-m_{i,k}}) \tag{3.1}$$

Intuitively, good explainability methods should identify the most important features to a prediction. Thus, when the most important features are occluded, the prediction difference should be large.

Consistency: The similarity of explanations under fixed conditions.

Let R be the number of runs, k be the number of top features selected by each explanation, $m_{i,k}^r$ be the importance mask for run r, and $\sum_{r=1}^{R} I(A_j^r = 1)$ be the number of times feature j was selected by the r explanations.

$$Consistency = \frac{1}{|V|}\sum_{i=1}^{|V|}\frac{1}{R}\sum_{r=1}^{R}\frac{\sum_{j\in m_{i,k}^r}\sum_{r=1}^{R} I(A_j^r = 1)}{R\cdot k} \tag{3.2}$$

Intuitively, a good explanation should be consistent across numerous runs using the same GNN model, dataset, and method parameters. The higher the consistency, the better – the maximum value of 1 is achieved when each of the R explanations chooses the same top k features and the minimum value of $1/R$ is achieved when each explanation chooses disjoint top k features.

Data Stability: The difference in explanation once input features are slightly perturbed,

conditional on class label remaining unchanged.

Let $X'_i = X_i + \varepsilon$ s.t. $\varphi(X'_i) = \varphi(X_i)$. Then let $A_i = E(f'_i, G)$ and $A'_i = E(f'_i, G)$ for some explainer E and fixed graph G.

$$DataStability = \frac{1}{|V|} \sum_{i}^{|V|} \frac{< A_i, A'_i >}{||A_i||\,||A'_i||} \tag{3.3}$$

Intuitively, a good explanation should be invariant to small changes that do not affect the predictions of the model. The degree of change in the explanation should be small.

Fairness: The presence of a sensitive attribute in the explanation (if a prediction is biased). The similarity of predictions using the generated explanations as a feature mask when a sensitive attribute is perturbed (if a prediction is unbiased). A prediction's bias is determined by whether or not the prediction changes when the sensitive attribute of a node is flipped.

Let $a \in \{0, 1\}$ be the sensitive attribute we aim to protect, $y_{i,a}$ be the original prediction for node i with sensitive attribute a from the GNN, and $y_{i,\tilde{a}}$ be the prediction for node i with the flipped sensitive attribute $1 - a$.

$$Fairness(biased) = \frac{1}{\sum_i 1(y_{i,a} \neq y_{i,\tilde{a}})} \sum_{i: y_{i,a} \neq y_{i,\tilde{a}}} 1(a \in m_{i,k}) \tag{3.4}$$

$$Fairness(unbiased) = \frac{1}{\sum_i 1(y_{i,a} = y_{i,\tilde{a}})} \sum_{i: y_{i,a} = y_{i,\tilde{a}}} 1(y_{i,a}^{m_{i,k}} = y_{i,\tilde{a}}^{m_{i,k}}) \tag{3.5}$$

Intuitively, an explainability method should be as biased as the underlying GNN. A fair explainability method should accurately convey the amount of bias in the underlying GNN model's predictions.

Scalability: The feasibility of generating accompanying explanations with model predictions in at-scale machine learning use-cases.

Let $t_{pred}(n),\ t_{expl}(n)$ be the time it takes to generate n predictions and explanations respectively.

$$Scalability(n) = \frac{t_{expl}(n)}{t_{pred}(n)} \tag{3.6}$$

Intuitively, an explainability method should be able to generate large amounts of explanations along with predictions under strict latency requirements (i.e., SLA targets in the milliseconds range)[17,16,22].

Many of the most impactful machine learning use-cases are deployed at-scale under strict latency requirements. For example, over 80% of machine learning inference cycles on Facebook's datacenter fleet are devoted to recommendation ranking under specific latency targets[17,16,34,22]. In these cases, in order to generate an accompanying explanation to a prediction, the time that it takes to formulate the explanation must be considered. We plan on evaluating this by first characterizing the FLOPs (floating point operations) and memory accesses associated with each explainability method then investigating the amount of ex-

ploitable parallelism for each explainability method.

3.3 GNN explanation methods

1. **GNNExplainer** is a *perturbation-based* method that has three steps. First, a initial mask is chosen for node features and treated as a trainable variable. Second, the mask is combined with the node. Finally, the mask is optimized to maximize mutual information between the original node and the masked node.
2. **GraphLIME** is a *surrogate method* which extends LIME to GNNs. GraphLIME has three steps. First, *N*-hop neighboring nodes of a target node are considered as the local region for explanation. Second, Hilbert-Schmidt Independence Criterion (HSIC) Lasso is utilized as the local surrogate model. Third, feature importance is assigned based on magnitudal weights of HSIC Lasso coefficients.
3. **Sensitivity Analysis (SA)** utilizes squared gradients as measures of feature importance. Utilizing backpropagation, input features are used as targets instead of weights.

3.4 Datasets

	Classes	Features	Nodes	Edges	Average Node Degree	Edge Density
Cora	7	1433	2708	10556	3.9	0.00287
Pubmed	3	500	19717	88648	4.5	0.000456
NBA	2	96	403	16570	41.12	0.204

3.4.1 GNN Models:

We analyze three canonical GNNs: Graph Convolutional Networks (GCNs), GraphSAGE, and Graph Attention Networks (GATs). The first two leverage the graph convolution op-

erator for aggregation – GCNs spectrally and GraphSAGE spatially; GATs leverage the attention mechanism for aggregation.

3.5 Experimental results

3.5.1 Fidelity Analysis

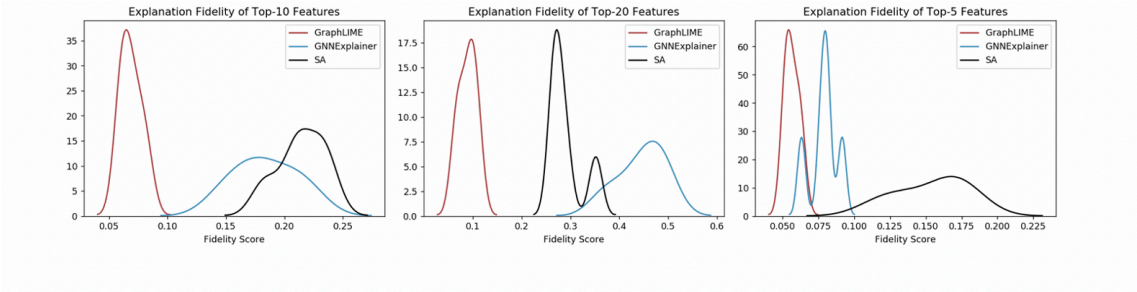

Figure 3.1: Fidelity of different explainability methods on Pubmed and Cora. GNNExplainer is more successful on large k (i.e., # of features in explanation) while SA is more successful for small k.

As shown in the KDE plots in Figure 3.1, SA performs well for low values of k, while GNNExplainer performs well for high values of k. GraphLIME tends to have the poorest performance, most especially for complex datasets with high number of nodes and number of features per node. The intuition behind the difference in performance for SA versus GNNExplainer lies in the importance of neighboring node features. That is, for low values of k, the features of the target node themselves are the key attribution to importance magnitudes. However, as we increase k and consider the importance of other features in the target node, how neighboring node features and specifically the hidden layer embedding representations

of those features contribute to target features in a larger way.

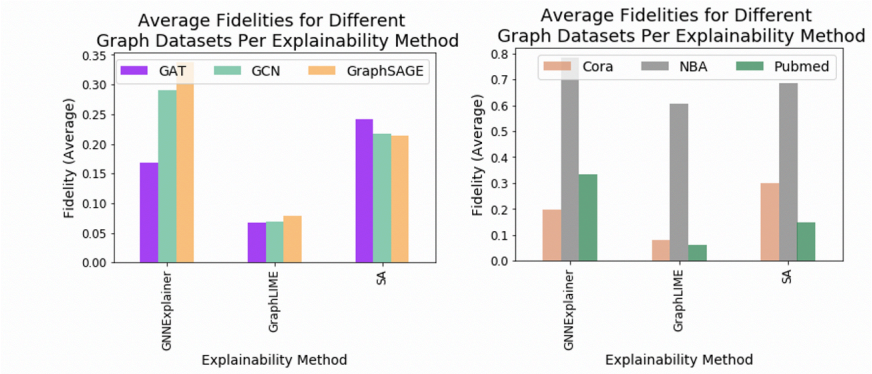

Figure 3.2: Fidelity of different explainability methods across datasets **(Left)** and GNN models **(Right)**.

Next, we consider dataset based analysis. As seen in Figure 3.2 (Right), SA has higher fidelity on datasets with higher number of node features but lower node degree (i.e., Cora). Meanwhile, GNNExplainer tends to perform better on datasets with higher node degrees and node features (i.e., NBA, Pubmed). The intuition is that GNNExplainer takes better account of spatial information in a N-hop neighborhood. Thus, with higher node degree, GNNExplainer can better account for relationships between nodes. Meanwhile SA's objective function is based solely on a target node's input feature. Hence it is better able to parse importance when there is a high number of node features. Similarly as seen in Figure 3.2 (Right), when comparing performance across models, GNNExplainer performs well on spatial models like GraphSAGE while SA performs better on spectral models like GCN. In a spectral graph convolution, we perform an Eigen decomposition of the Laplacian Matrix of the graph while in spatial we use N-hop to aggregate nodes.

Finally, GraphLIME's overall low fidelity can be attributed to a loss in information using its local approximation and surrogate model. That is, not only is the N-hop restrictive, but HSIC-Lasso simply cannot approximate the spatial-based and non-linear GNN target function for complex datasets. Interestingly enough, we see that there is no clear correlation between model accuracy and explanation fidelity As expected, our explanation fidelities increase when we are able to pick more features in our explanations.

3.5.2 Fairness Analysis

Biased			
	Top 5	**Top 10**	**Top 20**
GraphLIME	0.664	0.885	0.915
GNNExplainer	0.488	0.518	0.554
SA	0	0.028	0.133

Unbiased			
	Top 5	**Top 10**	**Top 20**
GraphLIME	0.750	0.675	0.662
GNNExplainer	0.852	0.817	0.770
SA	0.996	0.956	0.948

Table 3.1: Overall Fairness Metrics

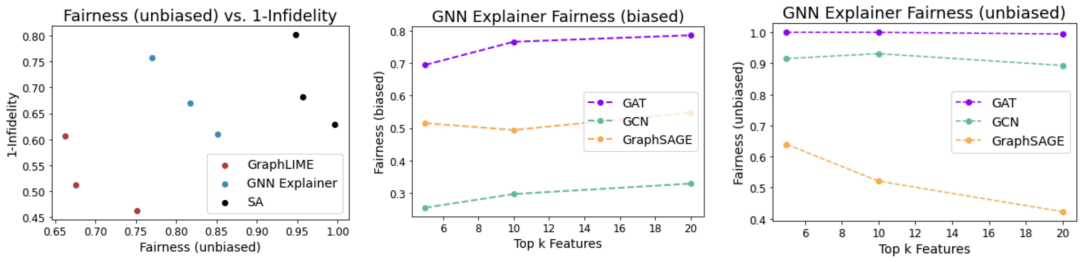

Figure 3.3: a) Scatterplot of average fairness and 1-infidelity scores, b) GNN Explainer fairness scores (biased), c) GNN Explainer fairness scores (unbiased).

From Figure 3.3, we see a positive correlation between fairness and fidelity when the prediction is unbiased. Intuitively, this makes sense because when the model is unbiased, fidelity becomes a proxy for fairness; the sensitive attribute is not used to make the predic-

tion, and thus the explanation's bias is dependent on its faithfulness to the model. Thus, we observe that Sensitivity Analysis, shown in Table 3.1 and in green in Figure **??**, has both the highest average fidelity and fairness (unbiased) scores.

Next, Table 3.1 reveals that GraphLIME is the fairest method for biased predictions, i.e. it most frequently identified the sensitive attribute as an important feature. This can be attributed to GraphLIME's use of both the prediction's neighborhood and node feature importance targets. On one hand, GNN Explainer exploits graph structure through the N-hop neighborhood, but focuses less on finding useful features, thus hindering its ability to identify the sensitive attribute. On the other hand, SA solely focuses on node features as targets, but does not explicitly learn a latent representation of the node's neighborhood. This prevents a sufficient explanation because similar people tend to be connected to each other, and the GNN message passing mechanisms exacerbate those dependencies[33,29,57].

Lastly, we hypothesize that synergies between explainability method and GNN model result in high fairness. As shown in the first two graphs of figure 3.3, GNN-Explainer and GATs perform extremely well together, as both methodologies involve localizing through edge-wise mechanisms. GNN-Explainer learns an edge mask to find a local subgraph to explain predictions, and GATs' aggregation function similarly attends over neighborhoods and learns different importance weights to assign to different neighbors. Thus, two are able to complement each other and find the fairest explanations.

3.5.3 Consistency Analysis

Cora			
	Top 5	**Top 10**	**Top 20**
GraphLIME	1.0	1.0	1.0
GNNExplainer	0.429	0.561	0.682
SA	1.0	1.0	1.0

Pubmed			
	Top 5	**Top 10**	**Top 20**
GraphLIME	1.0	1.0	1.0
GNNExplainer	0.467	0.572	0.709
SA	1.0	1.0	1.0

Table 3.2: Overall Consistency Metrics

As shown in Table 4.1, SA and GraphLIME demonstrate perfect consistency across both Cora and Pubmed and all top k features, whereas GNNExplainer performs significantly worse across the board[†]. Regarding SA, we see complete consistency due to its deterministic back-propagation algorithm; given a fixed seed, model, dataset, and target node, the gradient and weight update computations are independent from the number of times the method is run. Similarly, GraphLIME exhibits perfect consistency, as it employs the kernelized nonlinear explanation model HSIC LASSO and optimizes with least angle regression, both of which perform deterministically under fixed conditions mentioned previously. Lastly, we see that GNNExplainer demonstrates the poorest consistency across both datasets and all top k features. Different from GraphLIME and SA, GNNExplainer's objective function jointly learns graph structural and node feature information to generate an explanation. Thus, variability within edge masks across runs significantly affects the consistency of the node feature masks learned. In the future, we plan to conduct further analysis

[†]The same conclusion holds for different datasets and models

on the consistency of edge mask learning and how this affects the node feature mask.

Interestingly, when analyzing consistency across the top k features, we see that GNNExplainer's performance improves as the number of features selected increased. Because our consistency metric does not compare the weights assigned to each feature but instead the set of top k features chosen, this suggests that GNNExplainer is able to identify a consistent batch of features which are important to the prediction, but is not consistent in assigning the relative importance of the top features.

3.5.4 Scalability Analysis

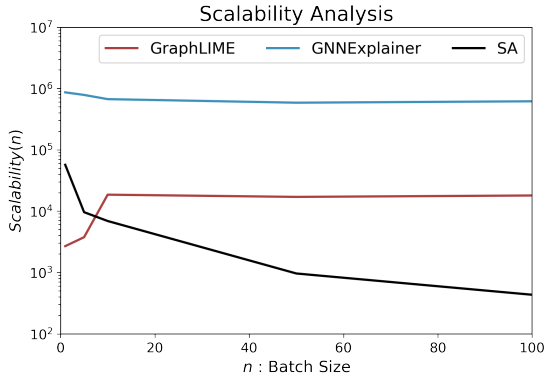

Figure 3.4: Even though off-the-shelf explainability methods are generally orders of magnitude slower than GNN inference, some are more amenable to parallelism.

Figure 3.4 depicts our observations on the feasibility of serving explanations at-scale along with predictions. Recall that $Scalability(n) = \frac{t_{expl}(n)}{t_{pred}(n)}$ is a ratio where larger values indicate longer relative times to generate explanations. We see that this ratio is often in the $10^3 \sim 10^6$ range. Between explainability methods, however, we see variations in time to

explanation: SA (black), which primarily relies on backpropagation to find gradients, is the fastest; GNNExplainer (blue), which requires an exploration phase to find the best edge mask, is slowest. Beyond direct measurements of latency to explanation, we see that SA is also more amenable to parallelism. That is, as we generate more explanations in a batch (x-axis), the average time per explanation decreases; intuitively, this is because backpropagation is easier to run in parallel comapred with exploration/perturbation (GNNExplainer) and training a surrogate model (GraphLIME).

4

Recommendation Modeling

In our objective of understanding how and whether the explainability of recommendation systems provides benefits to the target user, we pursue two types of experiment. The first consists of traditional offline experiments in which pre-determined, mathematical metrics are deployed to measure the effectiveness of our methods. We cross-apply many of the same metrics present in the graph neural network section. In the online experiments, we survey a

representative population of target users to understand how, if at all, explainability impacts effectiveness of recommendation systems.

4.1 Recommendation models

We utilize two types of neural-based recommendation systems to generalize trends in offline experimentation. We select these models as they are considered state-of-the-art and some of the most widely used models in the field[64].

Neural Collaborative Filtering:

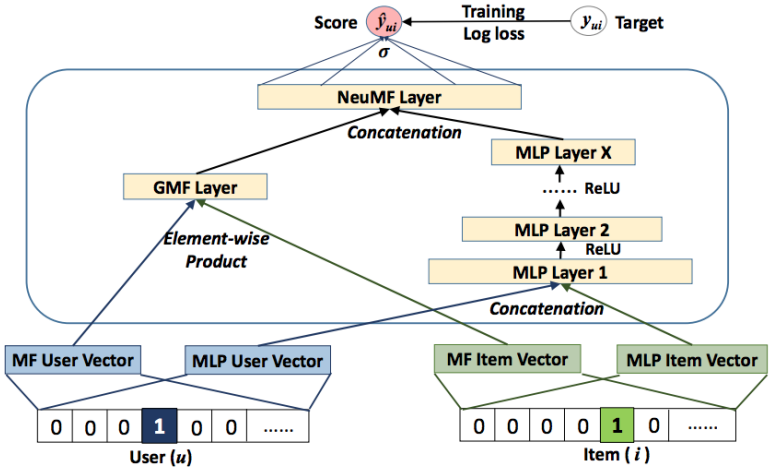

Figure 4.1: Illustration of neural collabrative filtering model architecture

The main benefit of a neural collaborative filtering model as compared to a standard matrix factorization models comes from the ability to model non-linear relationships between the item and user latent vector. That is, while standard matrix factorization simply utilizes

an inner product between the two to determine predicted rankings, neural collaborative filtering can extend this relationship to any functional form utilizing a multilayer perceptron [20].

The first layer of the model serves as the input. The input consists of two feature vectors $\mathbf{v}_u^U$ and $\mathbf{v}_i^I$ that describe user u and item i, respectively; they can be customized to support a wide range of modelling of users and items. Above the input layer is the embedding layer; it is a fully connected layer that projects the sparse representation to a dense vector. The obtained user (item) embedding can be seen as the latent vector for user (item) in the context of latent factor model. Following, neural collaborative filtering consists of two side-by-side set of layers. The first is a generalized matrix factorization section and the next is a multilayer perceptron section. The output of both sets of layer are then concatenated and fed into one last fully-connected layer. The final output layer is the predicted score $\hat{y}_{ui}$, and training is performed by minimizing the pointwise loss between $\hat{y}_{ui}$ and its target value y_{ui}. Thus, NCF's predictive model is formulated as:

$$\hat{y}_{ui} = f\left(\mathbf{P}^T\mathbf{v}_u^U, \mathbf{Q}^T\mathbf{v}_i^I \mid \mathbf{P}, \mathbf{Q}, \Theta_f\right)$$

We now speak a bit more about the two sections of NCF. The GMF, simply aims to mimic a matrix factorization model. More formally, let the user latent vector $\mathbf{p}_u$ be $\mathbf{P}^T\mathbf{v}_u^U$ and item latent vector $\mathbf{q}_i$ be $\mathbf{Q}^T\mathbf{v}_i^I$. We define the mapping function of the first neural CF

layer as:

$$\varphi_1(\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u \odot \mathbf{q}_i$$

where $\odot$ denotes the element-wise product of vectors. We then project the vector to the output layer:

$$\hat{y}_{ui} = a_{\text{out}}\left(\mathbf{h}^T(\mathbf{p}_u \odot \mathbf{q}_i)\right),$$

where a_{out} and $\mathbf{h}$ denote the activation function and edge weights of the output layer, respectively. Intuitively, if we use an identity function for a_{out} and enforce $\mathbf{h}$ to be a uniform vector of 1 , we can exactly recover the MF model.

Formally, the MLP model under our NCF framework is defined as

$$\mathbf{z}_1 = \varphi_1(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix}$$

$$\varphi_2(\mathbf{z}_1) = a_2\left(\mathbf{W}_2^T\mathbf{z}_1 + \mathbf{b}_2\right)$$

$$\ldots$$

$$\varphi_L(\mathbf{z}_{L-1}) = a_L\left(\mathbf{W}_L^T\mathbf{z}_{L-1} + \mathbf{b}_L\right)$$

$$\hat{y}_{ui} = \sigma\left(\mathbf{h}^T\varphi_L(\mathbf{z}_{L-1})\right)$$

where $\mathbf{W}_x$, $\mathbf{b}_x$, and a_x denote the weight matrix, bias vector, and activation function for the x-th layer's perceptron, respectively.

Wide & Deep Model:

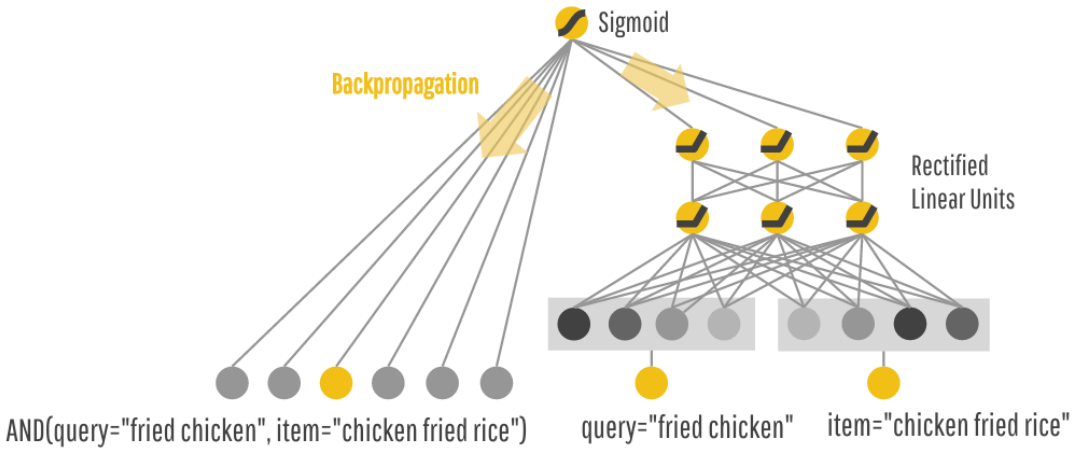

Figure 4.2: Illustration of Wide & Deep model architecture

Wide & Deep models are based on the idea that humans are able to generalize and both memorize. Thus, this model architecture utilizes a wide section to memorize and a deep section to generalize for recommendations[6].

The wide component is a generalized linear model of the form $y = \mathbf{w}^T\mathbf{x}+b$ where y is the prediction, $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ is a vector of d features, $\mathbf{w} = [w_1, w_2, \ldots, w_d]$ are the model parameters and b is the bias. The feature set includes raw input features and transformed features. One of the most important transformations is the cross-product transformation, which is defined as:

$$\varphi_k(\mathbf{x}) = \prod_{i=1}^{d} x_i^{c_{ki}} \quad c_{ki} \in \{0,1\}$$

where c_{ki} is a boolean variable that is 1 if the i-th feature is part of the k-th transformation

ϕ_k, and 0 otherwise. The deep learning component of the wide and deep model is equivalent to the MLP layer of a neural collaborative filtering model.

Ultimately, the wide component and deep component are combined using a weighted sum of their output log odds as the prediction, which is then fed to one common logistic loss function for joint training.

4.2 Explainability methods

For all of the models above, embedding layers play a distinctive role. That is, for the neural collaborative filtering model, all of the inputs are eventually fed into an embedding layer. This is because features such as *userid* or *itemid* are too sparse to be utilized in a meaningful way in the model. Similarly, while some of the inputs for the Wide & Deep model have intuitive meaning, such as the cross-features, the deep part of the model too contains an embedding layer. Thus, given the existence of the embedding layer as being instrumental in any deep-learning recommendation system, **explainability method tackling the problem must be able to handle this component**.

Most explainability methods, especially those that are post hoc, emphasize feature importance as the final goal. That is, many of the methods discussed in section 2.2, such as LIME and SHAP, focus on outputting coefficients that describe what inputs are the most influential. The difficulty in recommendation systems comes from the fact that there are

often very few inputs. For instance, neural collaborative filtering only has two inputs in the model: *userid* or *itemid*. It would be meaningless for LIME to say that a *userid* is more important than an *itemid* or vice versa. Moreover, it is not an individual item or user that influences the prediction of a recommendation, rather the patterns and sum of similar users and similar items. Thus, we must adapt our current understanding of post hoc explanation methods beyond their traditional application to input features.

Our proposal is that instead of applying our traditional post hoc explainability methods to the raw inputs of recommendation models, we apply them to the outputs of the embedding layer because embedding outputs represent traditional input features in other applications. These embedding vectors are often larger than just two inputs, so there can be meaningful comparison between which indices of the embedding vectors are more or less important.

Of course, the primary issue here is that embedding layers are inherently uninterpretable. They are a series of trained weights that do not correspond to features. Thus, we develop two novel methods to also explain embedding layers in a post hoc fashion. The combination of explaining what each weight of an embedding layer does and how the neural network utilizes the embedding layer creates a formula for us to fully explain black box recommendation systems.

4.2.1 Preliminaries

Consider the rating prediction problem where the input space $\mathcal{X}$ consists of a vector of users $U = [u_1, u_2, \ldots u_m]$ and set of items $I = [i_1, i_2, \ldots, i_n]$, and the output space $\mathcal{Y}$ is the set of possible ratings. Let $\mathcal{R} = \{z_1, z_2, \ldots, z_n\}$ be a set of observed user-item ratings, where $z_j = \left(u_j, i_j, y_j\right) \in \mathcal{X} \times \mathcal{Y}$. Without loss of generality, we define $L\left(z_j, \theta\right)$ as the loss function at $\left(u_j, i_j\right)$. The LFM is trained based on $\mathcal{R}$ and the model parameters $\hat{\theta}$ on convergence satisfies: $\hat{\theta} \stackrel{\text{def}}{=} \arg\min_\theta \frac{1}{n} \sum_{j=1}^{n} L\left(z_j, \theta\right)$. Problem. Finally, assume that for each user there is a corresponding vector of a_{u_j} of some fixed size s which corresponds to all aspects of a user where each entry is between 0 and 1. Similarly, a a_{i_j} of a fixed size t where each entry is between 0 and 1. All models will have a learned E_U and E_I of size $m \times d$ and $n \times d$ respectively where d is a pre-determined embedding size.

4.2.2 Methods to Explain the Latent Dimension:

To our knowledge, we propose two novel methods for post hoc explainability on latent dimension layers. Under each method, we separate based on two inputs. In our effort to create explainability methods that generalize to all recommendation systems, not just those with textual reviews, the first type of input is universally present in all recommendation models: neighboring items and users. That is, we explain user embeddings by referring to the most important users towards that explanation and explain item embeddings by pin-

pointing most important items. All collaborative filtering, and thus modern recommendation systems, contain user and item pair history.

The second type of input is also often present in most modern recommendation systems. These inputs are the traditional inputs to content filtering methods. We use user-level descriptive aspects such as age and gender to explain user embeddings. We use item-level descriptive aspects such as genre to explain item embeddings.

Ultimately, the goal in this stage is to align each individual or group of individual weights in the embedding layer to an important item/user or item aspect/user aspect.

Our problem statement is to utilize R along with the aspect vectors A_U and A_I to output a ranked vector V of size d of explanation users, items, user aspects, or item aspects. In the case of explanations for E_u, each input in the ranked vector can either be u_j or A_{u_j} and in the case of E_i each input in the ranked vector can either be u_j or A_{u_j}

When looking at neighbor-based approaches:

1. The first step in the surrogate model is to initialize an input matrix for our surrogate model. For each $u_j \in U$ we create a vector of size m. Thus the matrix M_u is of size $m \times m$ For each $i_j \in I$ we create a vector of size m. Thus, the matrix M_i is size $n \times n$. The matrix is initialized with zeroes.
2. We iterate through the training data or $\mathcal{R} = \{z_1, z_2, \ldots, z_n\}$. Beginning with some user u^*, we find all data here $z_j = (u^*, i_j, y_j)$. We make a list of every i_j corresponding to that user. Similarly in matrix, for every i^* corresponding to $z_j = (u_j, i^*, y_j)$, we make a list of all u_j.
3. Once again iterating through our training set, we find other users who have rated the same i_j or movies as the target user u^*. We take the average of their appearances and

update the corresponding user index in row u^* of M_u. We do a similar process for the item index in row i^* of M_i

4. We iterate through each index of E_u for a total of d target vectors where the target is the weight value in the embedding vector. We run d simple linear regressions of M_u as the independent values and the ith row of E_u. Similarly we run d simple linear regressions of M_i as the independent values and the ith row of E_i. Ultimately, we get a $2 * d$ lists of coefficients.

5. For each weight in the embedding layer, we select the top k coefficients. Thus, for the users, we can explain every index in the embedding layer by most influential users and vice versa for the item embedding layer.

When looking at aspect-based explanations as opposed to neighbor based explanations:

1. For each $u_j \in U$ we create a vector of size s. Thus the matrix M_u is of size $m \times s$ For each $i_j \in I$ we create a vector of size t. Thus, the matrix M_i is size $n \times t$. The matrix is initialized with zeroes.

2. For each row corresponding to a user in the M_u, we fill it with their attributes found in a_{u_j}. Similiar proccess occurs of M_i and a_{i_j}

3. We iterate through each index of E_u for a total of d target vectors where the target is the weight value in the embedding vector. We run d simple linear regressions of M_u as the independent values and the ith row of E_u. Similarly we run d simple linear regressions of M_i as the independent values and the ith row of E_i. Ultimately, we get a $2 * d$ lists of coefficients.

4. For each weight in the embedding layer, we select the top k coefficients. Thus, for the users, we can explain every index in the embedding layer by most influential user or item attributes.

Note, we only conduct explanations for the embedding layer for neural collabrative filtering. This is because Wide & Deep comes with pre-defined explanations for each input

already. Meanwhile the embeddings of the former model are inherently created by a black box.

4.2.3 Methods to Explain the Neural Models

Once we have an intuitive explanation method for each individual index in our embedding layer, we can then utilize adapted methods of traditional post hoc layers.

To apply LIME or KernelSHAP to deep-learning recommendation systems similar to neural collaborative filtering or Wide & Deep, we go through the following process:

1. Re-create your original model M with a new model M^* to delete input layers and thus create new input layers that take the embeddings as inputs
2. Similar take your test set of ratings and then iterate through each and replace the u_j and i_j with their corresponding values in the E_u and E_i matrix so you have R^*.
3. Run traditional LIME or traditional KernelSHAP on R^* and M^*

For vanilla gradients, we simply calculated the first-order gradients with respect to the embedding layers as opposed to the input.

4.3 Offline explainability metrics

In this section we will describe four general offline metrics we will utilize to measure the performance of both our novel explanation methods for latent dimensions as well as the adapted methods for explaining the neural layers. In general, we have five primary offline

metrics to judge the accuracy of both our latent explainability methods and neural explainability methods:

Fidelity: The difference between the original predictions and the new predictions after masking out important input features. Intuitively, good explainability methods should identify the most important features to a prediction. Thus, when the most important features are included, the prediction difference should be large.

For our latent method, we delete the majority of the instances that our method says are important from our train set. For instance, the most important user or item or deleting a group of users or items corresponding to the most important aspect. We then see the average ratio between the new embedding at index i and the old embedding.

For neural methods, we first mask out or set to zero the top k arguments of our embeddings as ranked by each of our explainability methods. We then calculate the mean squared difference between the new and old predictions across the test set.

Infidelity: The difference between the original predictions and the new predictions after masking out the least important input features. Intuitively, good explainability methods should be able to differentiate between the most and least important features. Thus, when the least important features are excluded, the prediction difference should be minimal.

For our latent method, we delete the majority of the instances that our method says are least important from our train set. For instance, the least significant user or item or delet-

ing a group of users or items corresponding to the least significant aspect. We then see the average ratio between the new embedding at index i and the old embedding.

For neural methods, we first mask out or set to zero the bottom k arguments of our embeddings as ranked by each of our explainability methods. We then calculate the mean absolute difference between the new and old predictions across the test set.

Consistency: The similarity of explanations under fixed conditions. Intuitively, a good explanation should be consistent across numerous runs using the same recommendation model, dataset, and method parameters.

For both types of explainability, we simply re-run the explainability method k times. We then calculate the cosine similarity between the new and original coefficients. We then take the average similarity across the k trials.

Data Stability: The difference in explanation once the input features are slightly perturbed, conditional on class label remaining unchanged. Intuitively, a good explanation should be invariant to small changes that do not affect the model's predictions. The degree of change in the explanation should be small.

For latent methods, we perturb k percent of the input ratings for each user. We then calculated the cosine similarity between the original coefficients and the new coefficients. For neural methods, we perturb k percent of the embedding values by some small ε. That is, we multiply the randomly selected indices by $1 + \varepsilon$. We then calculate the cosine similarity

between the original coefficients and new coefficients.

Scalability The feasibility of generating accompanying explanations with model predictions in at-scale machine learning use-cases. Intuitively, an explainability method should be able to generate large amounts of explanations along with predictions under strict latency requirements (i.e., SLA targets in the milliseconds range).

For both methods, we simply look at the time it takes to generate explanations in seconds.

4.4 Model & dataset paramters

For the neural collaborative filtering model, we tune over the following set of embedding layer sizes: 4, 8, 16, 32, 64, 128. Ultimately, 16 achieves the highest hit rate. This is likely due to the fact that the embedding layer achieves the most ability to represent the user and item history while not leading to overfitting. In addition to our embedding layer, we chose our MLP layers to have four dense layers of sizes: 64, 32 16, 8 with interspersing drop-out layers. For the Wide & Deep model, we tune over the same set of embedding layers. We also find that 16 achieves a similarly high hit rate. In addition to our embedding layer, we chose our MLP layers to have ten dense layers with interspersing dropout layers.

For our surrogate function to predict the importance of both neighbors and aspects, we utilize Ridge Regression with $\alpha = 1$. This minimized the mean squared error.

For our adapated LIME, for both models, we want to ensure that our explanation method achieves convergence on its output explanations. To test this, we rerun the method to have 250, 500, 1000, 2000, 4000 perturbed samples.

For KernelSHAP, we train our approximate function to predict feature contribution on the entirety of the train set, and thus convergence poses no problem. Similarly, for vanilla gradients, there is no sample size parameter and thus convergence does not need to be tested.

4.4.1 Offline Results: Neural Explanation Methods

Fidelity:

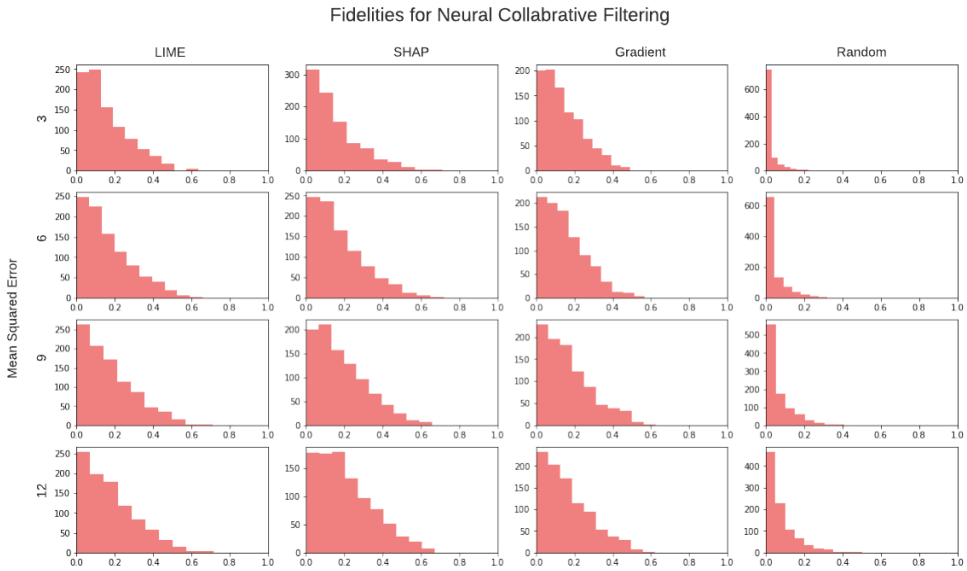

Figure 4.3: Neural collaborative filtering fidelities as measured by mean squared difference in prediction value for the top k ranked features based on various methods and for various k's

In general, for the neural collaborative filtering models, the SHAP method have the highest fidelity followed by the LIME and gradient methods in that order. LIME and SHAP tend to have similar fidelities across all k's, wheres the vanilla gradient method suffers comparatively. Regardless, all three methods tend to outperform much better than the random method, especially at lower k values.

We see LIME comparatively performs better on smaller k's, but SHAP tends to outperform on higher valued k's. This is likely because LIME's regression task is able to specifically pinpoint one or two major inputs whereas SHAP's problem formulation of coalitions is better able to identify larger groups of features that are important.

Infidelity:

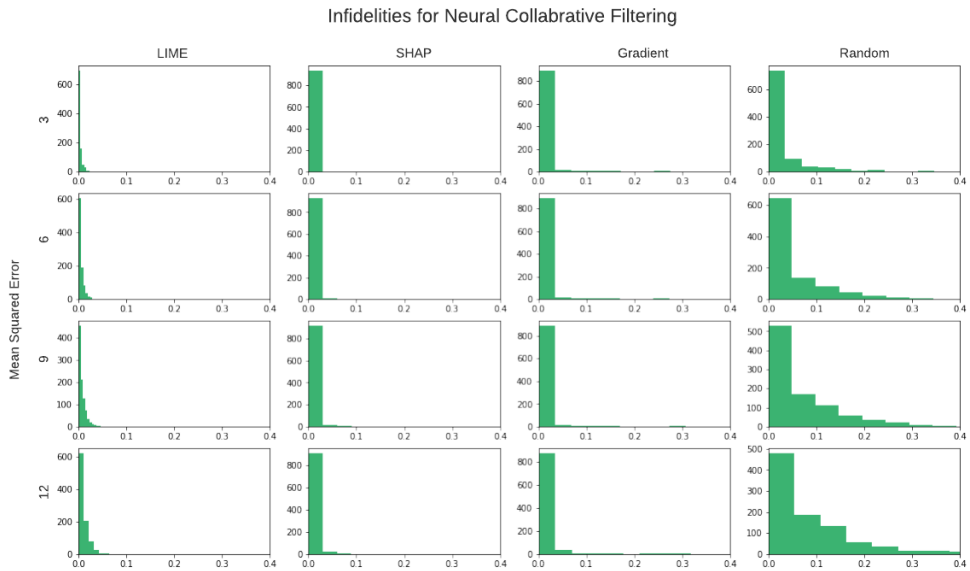

Figure 4.4: Neural collaborative filtering infidelities as measured by mean squared difference in prediction value for the top k ranked features based on various methods and for various k's

Similar trends are seen for infidelity in that the SHAP method tends to have the smallest change in performance for its lowest-ranked coefficients as compared to LIME and vanilla gradients. Vanilla gradients tends to be the closest to random, but all are quite distant from randomly chosen indices.

One may wonder why vanilla gradients performs much poorly compared to SHAP and LIME on both the fidelity and infidelity metrics. One reason may be vanishing gradients through the multi-layer perceptron layers. Another may be that vanilla gradients struggles due to the first-order approximation it relies on not being sufficient.

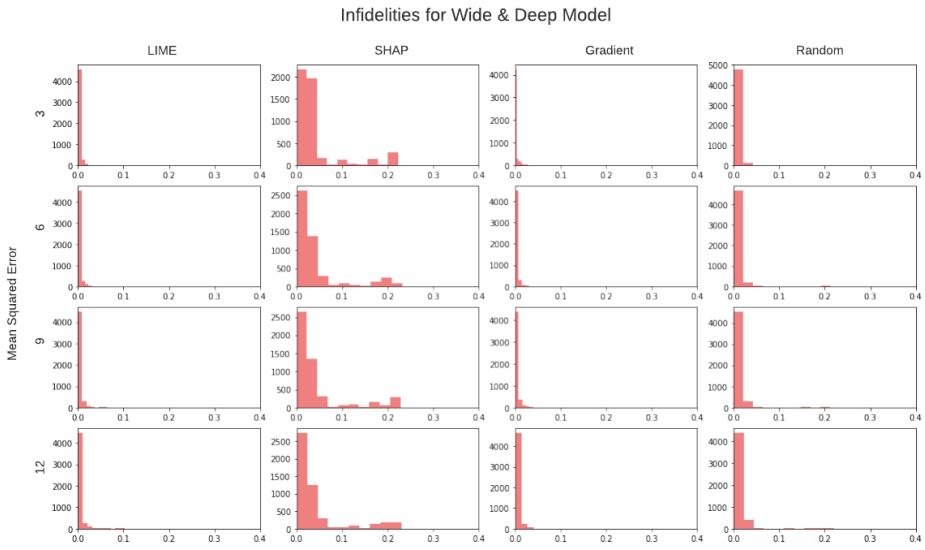

Figure 4.5: Wide & Deep fidelities as measured by mean squared difference in prediction value for the top k ranked features based on various methods and for various k's

For Wide & Deep fidelities, we get less promising results. While SHAP tends to perform

a significant margin above random, though less than its counterpart for neural collaborative filtering, both vanilla gradients and LIME tend to have more similar fidelity values to randomly selecting coefficients.

There are two possible explanations for why wide and deep tends to have lower fidelities than neural collaborative filtering. The first is that wide and deep has a relatively more complex neural architecture, making it more difficult for post hoc methods. The second is that wide and deep does not consider any feature that extracts historical data. That is, the only input in the model are demographic or item level information. This results in the model performing worse and likely having worse explainability.

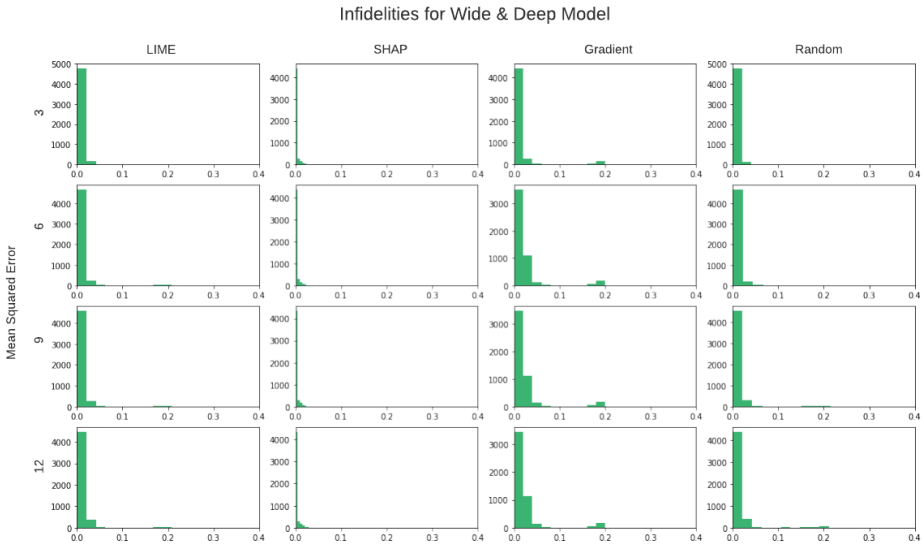

Figure 4.6: Wide & Deep infidelities as measured by mean squared difference in prediction value for the top k ranked features based on various methods and for various k's

For Wide & Deep infidelities, we see slightly better results in that every method performs above random, though again at a margin that is much smaller than neural collaborative filtering. It is interesting to note that in this case, SHAP performs much better than LIME than in the neural collaborative filtering case. This is likely because coalitions of inputs (i.e demographics and item-level features) are much more important than in the neural collaborative filtering case that does not consider demographics.

Consistency

Method	Average Consistency
LIME	0.997
SHAP	1.0
Vanilla Gradient	1.0

Table 4.1: Neural Collaborative Filtering Consistency

The neural collaborative filtering method tends to have extremely high consistency. There is no randomness in the method for vanilla gradients, hence the reasonable result that across all trials the output was the same. However, for SHAP, the high level of consistency points to why the method is likely so successful in fidelity. Finally, as expected, LIME has a consistency that is slightly less than one due to the inherent randomness of perturbing samples to create a local neighborhood. We can examine the consistency of LIME with greater depth with the boxplots below.

In the box plots, we see how LIME is converging to stable coefficients with larger number of perturbations. That is, both the average euclidean distance between the outputted

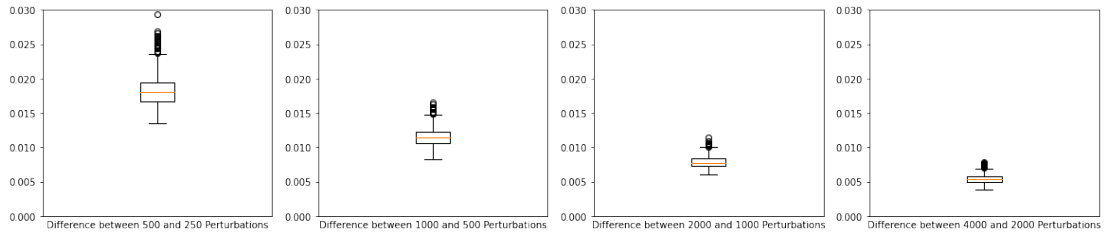

Figure 4.7: Box plots illustrating convergence for adapted LIME for neural collabrative filtering

coefficients and the spread as measured by interquartile range decreases with larger number of perturbations. At the end, we still see a nonzero difference, meaning that while LIME has almost converged, there will always be room for a non-perfect consistency unless we use theoretically infinite perturbations.

Method	Average Consistency
LIME	0.965
SHAP	1.0
Vanilla Gradient	1.0

Table 4.2: Wide & Deep Consistency

We see similar trends in consistency for Wide & Deep. LIME tends to have a slightly lower consistency. We can examine the boxplots for convergence to understand why this occurs.

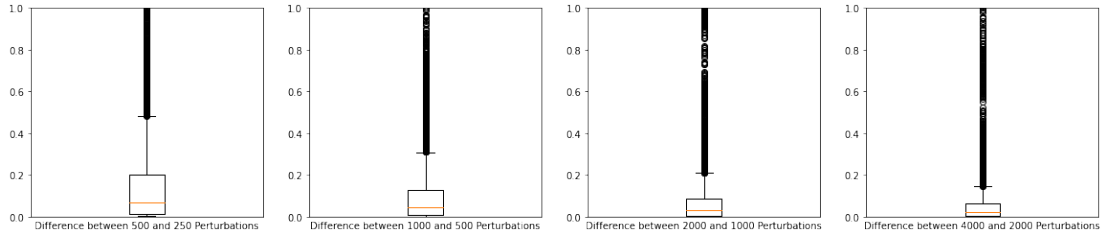

Figure 4.8: Box plots illustrating convergence for adapted LIME for Wide & Deep

As seen above, the convergence, while still occurring, is at a rate that is slower than the convergence of our LIME coefficients for neural collaborative filtering.

Data Stability

% Perturbed	LIME	SHAP	Vanilla Gradient
5	0.996	0.999	0.920
10	0.996	0.999	0.918
15	0.996	0.999	0.915
20	0.996	0.999	0.914

Table 4.3: Neural collaborative filtering data stability

Perturbation slightly of data tends to have minimal impact on SHAP and LIME as compared to vanilla gradients. Vanilla gradients tends to see a nontrivial amount of alteration in outputted coefficients based on small changes to the input. This is likely due to the fact that the neural network model is much more sensitive to small changes then the surrogate models that SHAP or LIME employ. That is, the gradient of the input likely change based on small changes to the input.

One could pinpoint this as a possible reason for low fidelity for vanilla gradients. That is, if vanilla gradients is so susceptible to small changes in the input, then its ability to generalize trends and patterns of explanations between similar data is likely worse.

For Wide & Deep, SHAP performs with near perfect data stability. LIME and vanilla gradients perform particularly worse. LIME's worse performance for wide & deep is in line with its comparatively worse performance on the fidelity metric as well.

% Perturbed	LIME	SHAP	Vanilla Gradient
5	0.942	0.999	0.910
10	0.942	0.999	0.905
15	0.942	0.999	0.895
20	0.942	0.999	0.894

Table 4.4: Wide & Deep data stability

Scalability

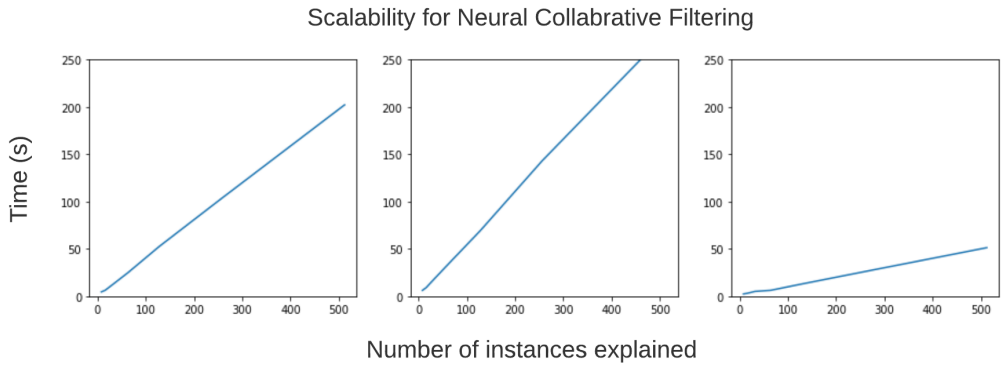

Figure 4.9: Neural collaborative scalability for LIME, SHAP, VanillaGrad respectively

The scalability trends point to LIME and SHAP having similar explanation generation times as compared to vanilla gradients. SHAP, however, in its need to also generate coalitions in addition to build a surrogate model, tends to take longer. One can then realize there may be an accuracy and scalibility trade-off ever-present in machine learning tasks. While LIME and SHAP tend to have better fidelity and data stability, they both are comparatively much more resource-intensive tasks. Regardless, all seem to have linear or $O(n)$ computational complexities.

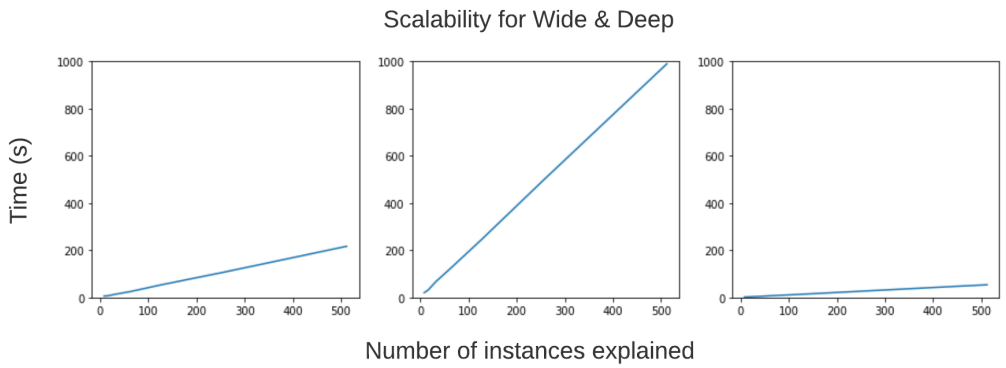

Figure 4.10: Wide & Deep scalability for LIME, SHAP, VanillaGrad respectively

In the case of Wide & Deep, the scalability of all tasks is still linear, although the coefficient for SHAP is much higher.

4.4.2 Offline Results: Latent explanation Methods

Fidelity & Infidelity

Neighbor-based:

Embedding Layer	Fidelity	Infidelity
GMF User	1.51	0.85
GMF Item	1.9	0.54
MLP User	1.45	0.86
MLP Item	1.05	0.95

Table 4.5: Neighbor-based embedding explanation fidelities and infidelities

Aspect-based:

In general the fidelities tend to be much better for the GMF layers than the MLP lay-

Embedding Layer	Fidelity	Infidelity
GMF User	1.11	0.91
GMF Item	1.10	0.98
MLP User	1.01	0.99
MLP Item	1.01	0.99

Table 4.6: Aspect-based embedding explanation fidelities and infidelities

ers. That is, it is easier for us to explain the GMF embedding layers than the MLP with our methods. This is likely due to two reasons. First, the GMF layer, in optimal, has a smaller embedding size which may be easier to explain. Second, the MLP embedding layer is ultimately connected a variety of neural network layers, which means that the embedding layer itself might not as naturally lend itself to interpretable inputs.

Moreover, we see that the neighbor-based method for all embedding layers has much higher fidelity than the aspect-based. This is logical as neural collaborative filtering is based on the idea that we can understand recommendations based on neighbors. Meanwhile, we don't specifically encode demographic or item-subject level information in our neural collabrative filtering model like the Wide & Deep model. Moreover, we also understand that the aspect-based approach may suffer due to unequal representation of various demographics.

For infidelities we observe similar trends. Indeed, as expected, the infidelities act almost as recriprocals of the fidelities.

Data Stability:

Embedding Layer	Data Stability Average	Data Stability Variance
GMF User	0.998	$1.471 * 10^{-6}$
GMF Item	0.924	$9.572 * 10^{-5}$
MLP User	0.995	$1.704 * 10^{-6}$
MLP Item	0.893	$3.922 * 10^{-4}$

Table 4.7: Data stability for neighbor-based explanations

Embedding Layer	Data Stability Average	Data Stability Variance
GMF User	0.927	$1.471 * 10^{-6}$
GMF Item	0.989	$9.572 * 10^{-5}$
MLP User	0.903	$1.704 * 10^{-6}$
MLP Item	0.981	$3.922 * 10^{-4}$

Table 4.8: Data stability for aspect-based explanations

Interestingly, we see higher data stability for user embedding layers in the neighbor-based approach, and higher for item embedding layers in the aspect-based approach. This would imply that adding additional item-user pairs has large impacts on the way we explain items, but doesn't have as large of an impact on groups of types of items. The vice versa applies for users. There are more item-level features, which likely means adding an additionl one does not have as large of an impact.

Scalability

We see the scalability trends for both aspect and neighbor methods have the same trends. The item explanations tend to take longer than users significantly. For the neighbor case this is likely because each item has more corresponding users than each user having more corresponding items. Moreover, for the aspect-based method, there are more item-level

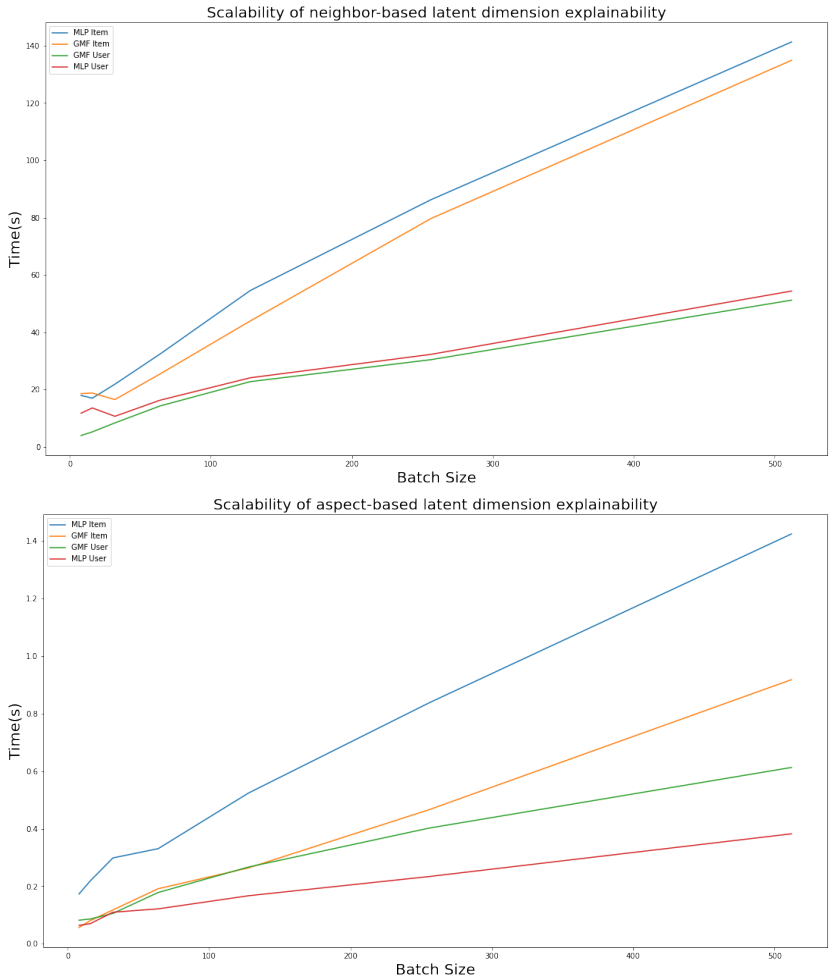

Figure 4.11: Top is scalability for neighbor-based explanations. Bottom is scalability for aspect-based explanations.

features than user-level features.

The scalability of the aspect method is clearly much less than the neighbor-based method. This is because for the neighbor-based method we have to search through an individual's neighbors but we do not have to complete this process with the aspect method.

However, both are still linear processes, which creates room for scalability.

4.5 Survey Motivation

Ultimately, human explainability is best defined by the target user or practitioner. Because explainability in AI is both a new field and because explainability for recommendation modeling is not widespread, we aim to test our methods utilizing the target users: consumers. Our survey aims to solve three primary goals:

1. RQ1: Do users prefer explainable recommendations as compared to non-explainable recommendations?
2. RQ2: What format of explanations do users most prefer? What post hoc explaianbility method do users most prefer?
3. RQ3: What is the relationship between recommendation accuracy and how much a target user cares about explainability of the recommendation" recommendations?

Our survey consists of three main subparts. The first is a demographic questionnaire. The purpose of the personal information questionnaire is to analyze any trends in explainability dependent on demographics. The second is profile selection in order for users to select candidates in our dataset that are similar to them. Finally, we go through the final section where we compare our explainability methods for generated recommendation movies.

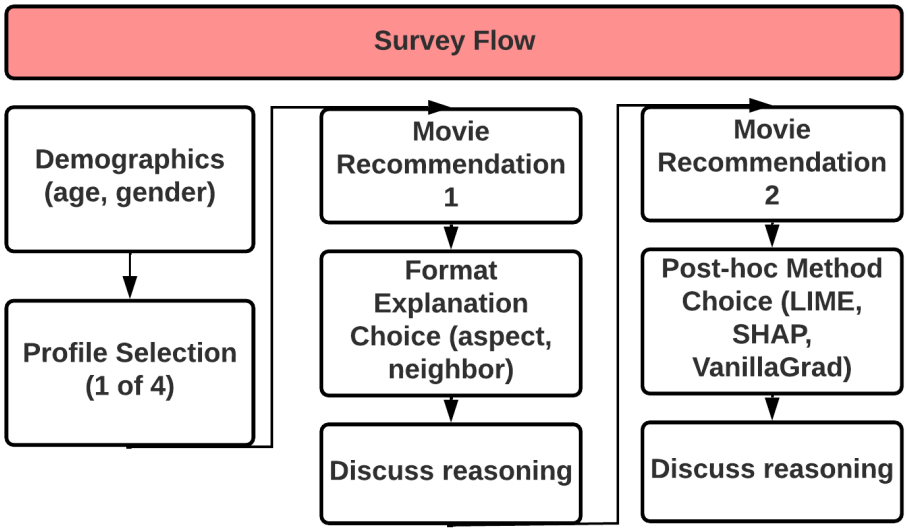

Figure 4.12: Figure depicts flow of survey

Demographics Section The demographic section aims to analyze trends based on age and gender. Moreover, we would like to ensure trends that are present in our study are readily extendable to a generalizeable population. As you can see for gender we allow three

options, understanding the exclusionary practice of the gender binary. For age, we allow a user to enter any number they desire. We only ask these two questions to minimize the privacy invasive nature of our survey. We understand that recent research by Dr. Latanya Sweeney has shown that adding additional questions such as zip code or occupation means we have the capacity to almost fully re-identify all participants in our study. Our final question of the three asks about technical experience to see if explainability in recommendation modeling is especially important to anyone or mainly those understanding the disagreement problem.

In the next section, survey users are given a list of four profiles each with four previously high-ranked movies. Each user is asked to select the profile that they most feel are similar to their own preferences.

Instead of letting users input or select their own movies, we use profiles for three primary reasons:

1. Users would have to input a large number of movies for the model to truly be accurate on them.
2. If users inputted their own movies or selves, we would have to not only re-train the model, but re-train every explanation method. This would be severe time bottleneck, making the survey less accessible.
3. Selecting movies would be especially difficult for younger users given that MovieLens datasets primarily consist of movies from before 2000.

To select the candidates that we profile from the Movielens dataset, we take the following

process. In the first step, we select the users that our model successfully includes their test movie rating in the test set. This is because we only want to select users that our model is accurate on. Next, of the approximately 700 potential candidates, we select the 100 which have the highest average fidelity over our LIME, SHAP, and Gradient methods. Once again, we only want to select users that have a baseline level of accuracy in our explanations as well as the model itself. Finally, from our remaining hundred, we select four that have diverse enough demographics. That is, each profile is randomly selected with the constraint that the age, occupation, and gender are different. From there, we run our user and every item in the MovieLens dataset through the neural collabrative filtering model trained in part 4 and select the top 2 predicted movies for each profile.

Comparison In our final part of the survey, we collect the important data. First and foremost, for each of the two movies that are recommended to the user, we ask if the user believes this is an accurate recommendation based on the profile they selected. It is important the user utilizes the profile data rather than their personal beliefs. This question aims to connect to RQ3.

The next question compares the three type of explainable formats. These are explaining based on a neighboring item, explaining based on a user aspect, and explaining based on an item aspect. We exclude a user neighbor as in this case that may be privacy invasive, though we understand there may be social networking applications in which a user neighbor would

be sufficient. We also include a no explanation choice. We then ask each user to explain their answer. This in hopes of answering both RQ1 and RQ2.

Following, we compare the explanations generated by LIME, SHAP, and VanillaGrad. For each post hoc method, we include an aspect explanation and a neighbor explanation. We once agian ask the user to explain their answer. This in hopes of answering both RQ1 and RQ2.

4.7 Survey results

Ultimately we were able to survey $n = 50$ members of the Harvard community and beyond to conduct our online testing. We believe that surveying the Harvard community would still be representative enough to understand the preferences of recommendation explanations as the survey in itself requires no technical expertise. Moreover, students at Harvard don't in themselves posess a statistical bias in regards to recommendation explainability.

Our demographics included 54% female, 42% male, and 2 non-gender binary. Our average age was 25 with a standard deviation of 8.

Comparison of Methods:

In total, we first compared the number of users who preferred any aspect-based explanation, neighbor-based explanation, or no explanation. Only 20.6% of users preferred no explanation as compared to 33.3% of users who preferred aspect-based explanations, and

45.0% of users who preferred neighbor-based explanations.

In terms of head-to-head comparisons or the first question of each section, we see there is a slight preference towards user aspects rather than item aspects. That is 18.1% of respondents preferred user-level aspects such as age or occupation as compared to the 15.1% that preferred item-level aspect information.

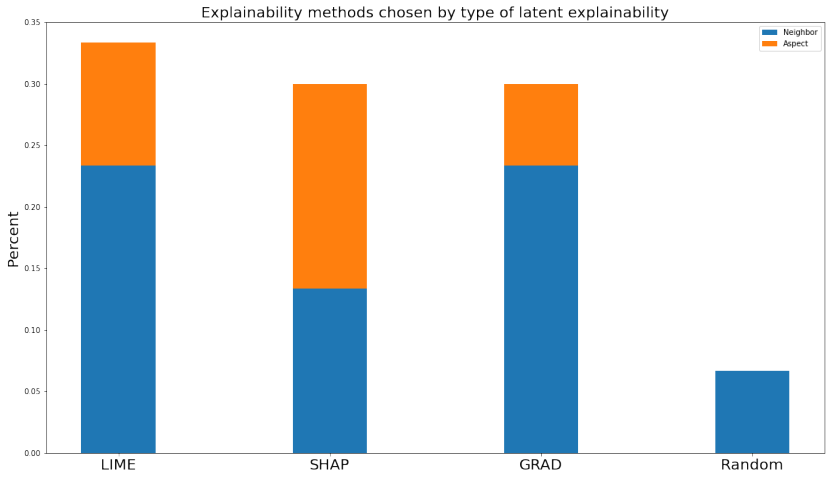

Figure 4.13: Figure shows breakdown of preference between neighbor and aspect latent explanations for each post hoc explanation

In general, when comparing the methods, the LIME method tends to gain more clicks as compared to a more equally clicked SHAP and VanillaGradient method. Moreover, while the neighbor explanation is preferred for LIME and Vanilla Gradients, SHAP sees a closer to equal split between the two latent explainability methods.

We then looked at relationship between what respondents thought of the accuracy of the recommendation with respect to the types of explanations they preferred. In general,

neighbor-based explanations were preferred when users perceived higher accuracy of recommendations while aspect-based recommendations were preferred with lower ranked recommendations. Intuitively, it may be easier to see an explanation is right if the ranking is right when it comes to similar movies. However, when a ranking seems wrong, a user may chalk that up to a demographic-level explanation they do not fully understand. In terms of trends when connecting post hoc method to recommendation quality, the most prominent is that VanillaGrad explanations are preferred at lower quality recommendations.

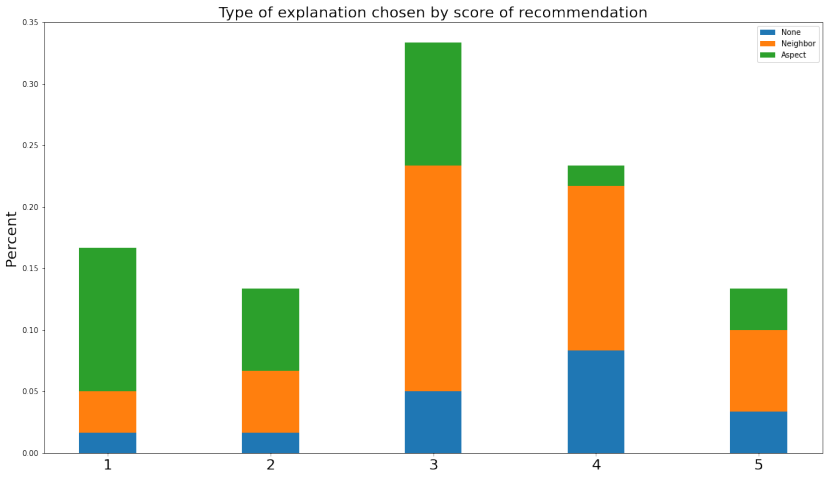

Figure 4.14: Figure shows breakdown of preference over neighbor or aspect explanations for various perceived recommendation qualities

When we asked users to explain why they clicked certain explanations, we see a few emerging themes.

First and foremost, users who chose not to receive an explanation did so because they were able to perceive incorrectness of the explanation methods. For instance, recommend-

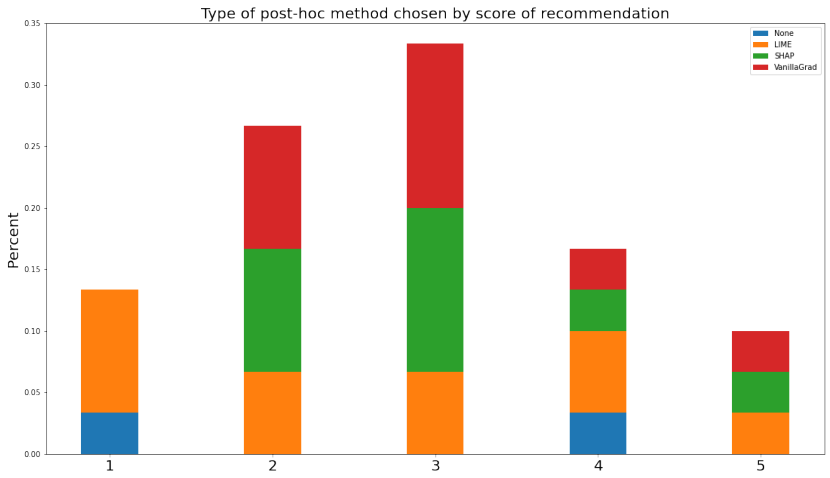

Figure 4.15: Figure shows breakdown of preference over LIME, SHAP, or VanillaGrad explanations for various perceived recommendation qualities

ing a movie because someone is age "30" even if they're profile is supposed to be in the "60's." This underscores the importance of resolving the disagreement problem. It is not just sufficient to explain, but correct explanations are vital. One example quotation was "I don't think the movie is similar to The Rock and Marvins room (2nd choice) and it is not a horror movie (4th choice). I'm not an artist or an engineer (3rd choice), so the 1st choice seemed most natural and likely to me."

Second, it is more easy to perceive an incorrect explanation in aspect-based methods. One example quotation was "it's like a preview of what people 25-31 like... even though I'm 69." In addition, others seem not to chose demographic information given that it may be privacy-invasive as some users claimed they liked explanations that lacked "demographic information."

Third, many view the neighbor-based explanations as algorithmic or trustworthy. One example quotation included "a rom-com seems like an amalgamation of most of the movies in that profile, even though I don't know if I'll like this one particularly." Another example quotation included "I would probably not like the movie, but I would be more accepting if it was supposed to be similar to other movies I have seen."

Thus, ultimately we achieve the following answers. For research question 1, we do find a significant difference between recommendations with explainability and recommendations without explainability. It is clear users prefer explainable recommendations. For research question 2, we find users prefer neighbor-based explanations generated by LIME. Finally, for research question 3, we find in both the format of the explanation and the post hoc method, there are significant trends between quality of recommendation and preferred explanation.

5

Conclusion

5.1 The Disagreement Problem conclusions

We introduced and studied the *disagreement problem* in explainable ML. More specifically, we formalized the notion of disagreement between explanations, analyzed how often such disagreements occur in practice, and how practitioners resolve these disagreements. We conducted interviews with data scientists to understand what constitutes disagreement

between explanations, and introduced a novel quantitative framework to formalize this understanding. We then leveraged this framework to carry out a rigorous empirical analysis with four real-world datasets, six state-of-the-art post hoc explanation methods, and eight different predictive models, to measure the extent of disagreement between the explanations generated by various popular explanation methods. We also carried out an online user study with data scientists to understand how they resolve explanation disagreements. Our results indicate that state-of-the-art explanation methods often disagree in terms of the explanations they output, and worse yet, there do not seem to be any principled approach that ML practitioners employ to resolve these disagreements. For instance, 84% of our interview participants reported encountering the disagreement problem in their day-to-day workflow. Our empirical analysis with real world data further confirmed that explanations generated by state-of-the-art methods often disagree with each other, and that this phenomenon persists across various models and data modalities. Furthermore, 86% of our online user study responses indicated that ML practitioners either employed arbitrary heuristics (e.g., choosing a favorite method) or just simply did not know how to resolve the disagreement problem.

We shed light on a unique problem that poses a critical challenge to adopting post hoc explanations in practice and pave the way for several interesting future research directions. First, it would be interesting to systematically study the reasons behind the occurrence of

the explanation disagreement problem. Second, it would be interesting to propose novel approaches to address this problem. One way to do this is to come up with principled evaluation metrics which can help practitioners readily discern a reliable explanation from an unreliable one when there is a disagreement. Third, it would also be interesting to rethink the problem of explaining ML from scratch, and potentially develop a whole new set of algorithms that are built on a common set of guiding principles to avoid these kinds of disagreements. Lastly, it would be also be incredibly important to regularly educate data scientists and practitioners about state-of-the-art approaches (e.g., novel evaluation metrics) that can be used to resolve disagreements between explanations.

5.2 Graph neural network conclusions

Next, we presented present the first holistic set of metrics to evaluate GNN explainability method performance along critical real-world objectives. We then propose an evaluation framework which calculates these metrics along the unique decision boundaries generated by the underlying dataset and model.

Through our analysis, we discovered the effect of dataset properties on explanation performance, synergies between certain explainability methods and underlying GNN models, and overall trends and trade-offs across metrics based on the use case.

We were able to identify and resolve the disagreement probelm successfully by discussing

in what situations certain classes of explaianbility methods should be applied. Below, we are also able to utilize our set of metrics to explain how these models could be improved so that their disagreement is not as widespread.

We first present an innovation for GNNExplainer. From our empirical analysis, we discovered GNNExplainer's main weakness lies in consistency due to its convexity assumption. That is, because the loss is not convex, GNNExplainer finds many adequate local minimum rather than a global minimum. The innovation is to include an additional hyperparameter N_t that accounts for how many times one should run a GNNExplainer for a target node then take the average of the feature importance. This will create more consistent and better estimator of true feature importance, however scalability would be a major issue given GNNExplainer already slow attribution time. Indeed, GNNExplainer lacks the ability to parallelize operations making this computationally difficult for large values of N_t

We also present a new innovation for Sensitivity Analysis. From our empirical analysis, we discovered SA's weakness in that it does not explicitly learn a latent representation of a prediction's N-hop neighborhood. Thus, instead of SA's current method of taking the gradients w.r.t just one prediction, we propose taking the gradients w.r.t. numerous neighboring predictions, weighting them according to similarity of node features, and aggregating them for a final importance measure. Because this method now considers the predictions of a node's neighbors, SA may experience better fidelity and fairness. However, this comes

at a trade-off of stability and scalability, as small changes across neighbors may be amplified through aggregation and additional computation is required.

In terms of an innovation for GraphLIME, we realized from our empirical analysis that there were two limitations: 1) training on only the k-hop neighborhood and 2) using HSIC LASSO as the surrogate model. For the first point, we can also incorporate "important" global nodes (i.e., nodes with high edge degrees) in our neighborhood construction – this will allow GraphLIME explanations to also capture global characteristics of the graph dataset. Second, we can use models that are more akin to graph-tasks but still lightweight and interpretable. For example, we can use a one-layer GAT as the surrogate model. These modifications will increase fidelity at a cost of scalability (training more complex models takes more time).

5.3 Recommendation modeling conclusions

Following, we extended our list of holistic metrics to evaluate recommendation model explainability. In our pursuit of testing and evaluating recommendation post-hoc explainability, we formulated recommendation explainability as a two-step process: latent dimension explainability and network explainability. For the former, we proposed a completely new methodology and for the latter we adapted current state-of-the art methods.

We conducted the first user study testing recommendation model explainability. We

discovered that users unequivocally prefer recommendations with explanations and began to understand the format and types of explanations that they prefer.

Ultimately, our goal was to resolve the disagreement problem. While a resolution to the disagreement problem will always be application-specific, we propose that those who use recommendation models similar to neural collabrative filtering utilize a combination of neighbor-based explanations for latent dimensions combined with an adapted post-hoc LIME. In general, this results in high user clicks, high offline fidelity and data stability, and scalability that is useful for real-world applications.

However, for models that are similar to Wide & Deep, we propose using the post-hoc method of SHAP. This is because the fidelity for LIME and VanillaGrad tends to be concerning low. Because only demographic-level information is inputted into many Wide & Deep models, we also propose the innovation of encoding historical user ratings as well. This will likely result in a better latent dimension explanation method through the neighbor-based method.

5.4 Future work

In this work, we have identified a critical disagreement problem in the literature on explainable machine learning, and have proposed strategies to resolve this problem for graph neural networks. Going forward, we plan to propose novel methods to address the disagree-

ment problem for different classes of models (e.g., recurrent neural networks) and domains (e.g., natural language processing). We also hope to propose new metrics which are more aligned with practical use cases to evaluate post hoc explanations. We plan to expand our metrics to incorporate the notions of fairness and scalability, and potentially add other metrics such as privacy.

References

[1] Ahmed, K. B., Bouhorma, M., & Ahmed, M. B. (2014). Age of big data and smart cities: Privacy trade-off. *CoRR*, abs/1411.0087.

[2] Baldassarre, F. & Azizpour, H. (2019). Explainability techniques for graph convolutional networks. *CoRR*, abs/1905.13686.

[3] Barenstein, M. (2019). Propublica's compas data revisited.

[4] Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J. M., & Eckersley, P. (2020). Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (pp. 648–657).

[5] Chen, X., Qin, Z., Zhang, Y., & Xu, T. (2016). Learning to rank features for recommendation over multiple categories. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16 (pp. 305–314). New York, NY, USA: Association for Computing Machinery.

[6] Cheng, H., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., & Shah, H. (2016). Wide & deep learning for recommender systems. *CoRR*, abs/1606.07792.

[7] Cinus, F., Minici, M., Monti, C., & Bonchi, F. (2021). The effect of people recommenders on echo chambers and polarization. *CoRR*, abs/2112.00626.

[8] Costa, F., Ouyang, S., Dolog, P., & Lawlor, A. (2017). Automatic generation of natural language explanations. *CoRR*, abs/1707.01561.

[9] Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16 (pp. 191–198). New York, NY, USA: Association for Computing Machinery.

[10] Doshi-Velez, F. & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *CoRR, abs/1702.08608*.

[11] Dua, D. & Graff, C. (2017). UCI machine learning repository.

[12] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2), 303–338.

[13] Gao, Y., Li, Y.-F., Lin, Y., Gao, H., & Khan, L. (2020). Deep learning on knowledge graph for recommender system: A survey.

[14] Glauber, R. & Loula, A. C. (2019). Collaborative filtering vs. content-based filtering: differences and similarities. *CoRR*, abs/1912.08932.

[15] Greenwell, B. M., Boehmke, B. C., & McCarthy, A. J. (2018). A simple and effective model-based variable importance measure. *ArXiv*, abs/1805.04755.

[16] Gupta, U., Hsia, S., Saraph, V., Wang, X., Reagen, B., Wei, G.-Y., Lee, H.-H. S., Brooks, D., & Wu, C.-J. (2020). Deeprecsys: A system for optimizing end-to-end at-scale neural recommendation inference.

[17] Hazelwood, K., Bird, S., Brooks, D., Chintala, S., Diril, U., Dzhulgakov, D., Fawzy, M., Jia, B., Jia, Y., Kalro, A., Law, J., Lee, K., Lu, J., Noordhuis, P., Smelyanskiy, M., Xiong, L., & Wang, X. (2018). Applied machine learning at facebook: A datacenter infrastructure perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (pp. 620–629).

[18] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

[19] He, X., Chen, T., Kan, M.-Y., & Chen, X. (2015). Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15 (pp. 1661–1670). New York, NY, USA: Association for Computing Machinery.

[20] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. (2017). Neural collaborative filtering. *CoRR*, abs/1708.05031.

[21] Hong, S. R., Hullman, J., & Bertini, E. (2020). Human factors in model interpretability: Industry practices, challenges, and needs. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW1), 1–26.

[22] Hsia, S., Gupta, U., Wilkening, M., Wu, C. J., Wei, G. Y., & Brooks, D. (2020). Cross-stack workload characterization of deep recommendation systems. In *2020 IEEE International Symposium on Workload Characterization (IISWC)* (pp. 157–168).

[23] Huang, Q., Yamada, M., Tian, Y., Singh, D., Yin, D., & Chang, Y. (2020). Graphlime: Local interpretable model explanations for graph neural networks.

[24] Jelodar, H., Wang, Y., Yuan, C., & Feng, X. (2017). Latent dirichlet allocation (LDA) and topic modeling: models, applications, a survey. *CoRR*, abs/1711.04305.

[25] Kaur, H., Nori, H., Jenkins, S., Caruana, R., Wallach, H., & Wortman Vaughan, J. (2020). Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1–14).

[26] Kipf, T. N. & Welling, M. (2017). Semi-supervised classification with graph convolutional networks.

[27] Lage, I., Chen, E., He, J., Narayanan, M., Kim, B., Gershman, S., & Doshi-Velez, F. (2019). An evaluation of the human-interpretability of explanation. *CoRR, abs/1902.00006*.

[28] Lakkaraju, H. & Bastani, O. (2020). " how do i fool you?" manipulating user trust via misleading black box explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society* (pp. 79–85).

[29] Li, Q., Han, Z., & Wu, X.-M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning.

[30] Lundberg, S. M. & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (pp. 4765–4774).

[31] Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., & Zhang, X. (2020). Parameterized explainer for graph neural network.

[32] MacKenzie, I., Meyer, C., & Noble, S. (2013). How retailers can keep up with consumers.

[33] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2019). A survey on bias and fairness in machine learning.

[34] Naumov, M., Kim, J., Mudigere, D., Sridharan, S., Wang, X., Zhao, W., Yilmaz, S., Kim, C., Yuen, H., Ozdal, M., Nair, K., Gao, I., Su, B.-Y., Yang, J., & Smelyanskiy, M. (2020). Deep learning training in facebook data centers: Design of scale-up and scale-out systems.

[35] Nielsen, I. E., Dera, D., Rasool, G., Bouaynaya, N., & Ramachandran, R. P. (2021). Robust explainability: A tutorial on gradient-based attribution methods for deep neural networks. *CoRR*, abs/2107.11400.

[36] Peake, G. & Wang, J. (2018). Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '18 (pp. 2060–2069). New York, NY, USA: Association for Computing Machinery.

[37] Pope, P. E., Kolouri, S., Rostami, M., Martin, C. E., & Hoffmann, H. (2019). Explainability methods for graph convolutional neural networks.

[38] Pope, P. E., Kolouri, S., Rostami, M., Martin, C. E., & Hoffmann, H. (2019). Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10772–10781).

[39] Poursabzi-Sangdeh, F., Goldstein, D. G., Hofman, J. M., Vaughan, J. W., & Wallach, H. (2018). Manipulating and measuring model interpretability. *CoRR, abs/1802.07810*.

[40] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016a). "Why should I trust you?" Explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144).

[41] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016b). "why should I trust you?": Explaining the predictions of any classifier. In *Demo at the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

[42] Robnik-Sikonja, M. & Bohanec, M. (2018). Perturbation-based explanations of prediction models. In *Human and Machine Learning*.

[43] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015a). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 211–252.

[44] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015b). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 211–252.

[45] Saadati, M., Shihab, S., & Rahman, M. S. (2019). Movie recommender systems: Implementation and performance evaluation. *CoRR*, abs/1909.12749.

[46] Sanchez-Lengeling, B., Wei, J., Lee, B., Reif, E., Wang, P., Qian, W., McCloskey, K., Colwell, L., & Wiltschko, A. (2020). Evaluating attribution for graph neural

networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems*, volume 33 (pp. 5898–5910).: Curran Associates, Inc.

[47] Schnake, T., Eberle, O., Lederer, J., Nakajima, S., Schütt, K. T., Müller, K.-R., & Montavon, G. (2020). Higher-order explanations of graph neural networks via relevant walks.

[48] Schwartzenberg, C., van Engers, T. M., & Li, Y. (2020). The fidelity of global surrogates in interpretable machine learning.

[49] Seo, S., Huang, J., Yang, H., & Liu, Y. (2017). Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17 (pp. 297–305). New York, NY, USA: Association for Computing Machinery.

[50] Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning important features through propagating activation differences. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research* (pp. 3145–3153).: PMLR.

[51] Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations*.

[52] Singh, J. & Anand, A. (2018). Posthoc interpretability of learning to rank models using secondary training data. *CoRR*, abs/1806.11330.

[53] Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). Smoothgrad: Removing noise by adding noise. *CoRR*, abs/1706.03825.

[54] Sovrano, F., Vitali, F., & Palmirani, M. (2021). Making things explainable vs explaining: Requirements and challenges under the GDPR. *CoRR*, abs/2110.00758.

[55] Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In *International Conference on Machine Learning* (pp. 3319–3328).

[56] Vu, M. N. & Thai, M. T. (2020). Pgm-explainer: Probabilistic graphical model explanations for graph neural networks.

[57] Wang, H. & Leskovec, J. (2020). Unifying graph convolutional neural networks and label propagation.

[58] Wang, X., Chen, Y., Yang, J., Wu, L., Wu, Z., & Xie, X. (2018). A reinforcement learning framework for explainable recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)* (pp. 587–596).

[59] Wang, Z., Liu, M., Luo, Y., Xu, Z., Xie, Y., Wang, L., Cai, L., & Ji, S. (2021). Advanced graph and sequence neural networks for molecular property prediction and drug discovery.

[60] Wu, L., Quan, C., Li, C., Wang, Q., & Zheng, B. (2017). A context-aware user-item representation learning for item recommendation. *CoRR*, abs/1712.02342.

[61] Ying, R., Bourgeois, D., You, J., Zitnik, M., & Leskovec, J. (2019). GNN explainer: A tool for post-hoc explanation of graph neural networks. *CoRR*, abs/1903.03894.

[62] Yuan, H., Tang, J., Hu, X., & Ji, S. (2020a). Xgnn: Towards model-level explanations of graph neural networks.

[63] Yuan, H., Yu, H., Gui, S., & Ji, S. (2020b). Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445*.

[64] Zhang, S., Yao, L., & Sun, A. (2017). Deep learning based recommender system: A survey and new perspectives. *CoRR*, abs/1707.07435.

[65] Zhang, X., Zhao, J., & LeCun, Y. (2015a). Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 649–657.

[66] Zhang, X., Zhao, J. J., & LeCun, Y. (2015b). Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626.

[67] Zhang, Y. & Chen, X. (2018). Explainable recommendation: A survey and new perspectives. *CoRR*, abs/1804.11192.

[68] Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., & Ma, S. (2014). Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th International ACM SIGIR Conference on Research Development in Information Retrieval*, SIGIR '14 (pp. 83–92). New York, NY, USA: Association for Computing Machinery.