Addressing Stability in Classifier Explanations

Siavash Samiei*
University of Waterloo
ssamiei@uwaterloo.ca

Nasrin Baratalipour
Google
nasrinbaratali@google.com

Pranjul Yadav
Google
pranjulyadav@google.com

Amitabha Roy
Google
amroy@google.com

Dake He
Google
dkhe@google.com

***Abstract*—Machine learning based classifiers are often a black box when considering the contribution of inputs to the output probability of a label, especially with complex non-linear models such as neural networks. A popular way to explain machine learning model outputs in a model agnostic manner is through the use of Shapley values. For our use case of abuse fighting in digital advertisements, one primary impediment of using Shapley values in explanations was a problem of instability. Specifically, the instability problem manifests as explanations for the same example varying greatly due to random sampling in the algorithm. We found it useful to view this problem explicitly as Monte Carlo integration in the form of averaging the model output while varying only a subset of features in the example to be explained. In turn, this guides the number of samples needed to achieve a stable estimate of individual Shapley values and unlocked the use of Shapley value based explainers for our models as well as classifiers in general, including neural networks.**

***Index Terms*—Shapley values, model explanations, neural networks**

I. Introduction

Digital advertising [1] is the commercial backbone of the internet. It provides the means for purveyors of products and services to reach potential buyers in a targeted and efficient manner, while also providing revenue to publishers and creators. Unfortunately, due to its reach and scale, it also attracts abuse by bad actors aiming to serve malicious and undesirable contents to viewers of advertisements.

Advertising platforms such as Google Ads make extensive use of machine learning (including deep neural networks [2]) to fight abuse in the ads ecosystem. Fighting abuse effectively, requires understanding model decisions as well as drawing a clear link from features to model output. This is important for humans so that they may understand why an abuse fighting model might flag an entity and safely leverage the ability of machine learning models such as deep neural networks to mine patterns in high dimensional spaces of often thousands of features. To gain insight into model decisions and feature importance, we use explanation algorithms. We provide more details of a practical deployment of these explainers at scale in Section V.

Shapley value based explainers are a popular way to explain the output of classification models [3]. Shapely values originate from cooperative game theory [25] and have the useful additive property that they sum to the difference between the model output on a particular input and the expected output of the model over all possible inputs. Therefore they are an intuitive way to present contributions of individual features for human interpretation. The key challenge in computing Shapley values is to compute the model output for subsets of features. Training a new model for every subset would result in a number of models exponential in the number of features. A practical approximation is therefore often applied by substituting a random "background" value for missing features when computing Shapley values. This however leads to unstable explanations.

Figure 1, shows two explanations generated for the same example from the same neural network. We focus on the top nine features and group the contribution of the next fifteen features into a tenth bucket. The *set of top features, their rankings and relative contributions all vary widely across the two runs*, a problem we refer to as "instability". In these two particular runs the second highest contributing feature is different across the two and contribute in opposite directions to the classification. This in turn is detrimental to deploying Shapley value based explainers in practise.

The instability originates from computing model outputs given a subset of features by substituting "background values" for missing features. Figure 2, shows the output of the underlying neural network as we vary the value of a single feature (f_20) over 300 forward passes through the model, keeping the other features fixed. As is evident, the output can vary widely leading to a large variance in the estimation of the model output with the feature missing if we were to sample a single background value for the missing feature.

In this paper, we draw a simple and explicit link between estimating a model's output given a subset of features and Monte-Carlo integration [26], that holds in a model agnostic way. This has been hinted at before [21] but not explicitly studied in terms of estimating model output given missing features. We make the link explicit and study it specifically in the context of our deployment. We empirically show a reduction in variance in the estimate of a model's output given subsets of missing features for our models. We also show that the final explanations become more stable by defining and measuring

*Work done during an internship at Google

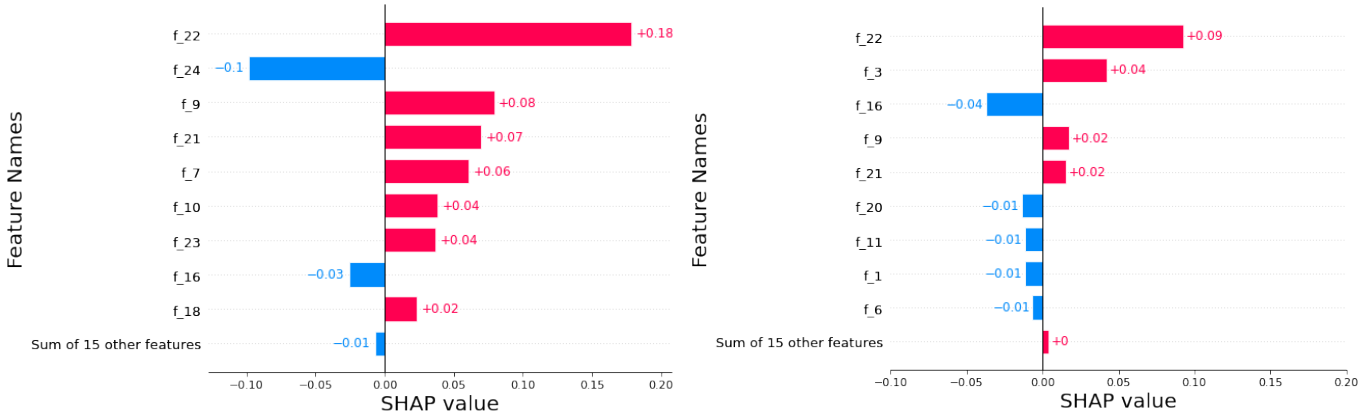

Fig. 1: Instability in two explanations generated for the same example from a neural network

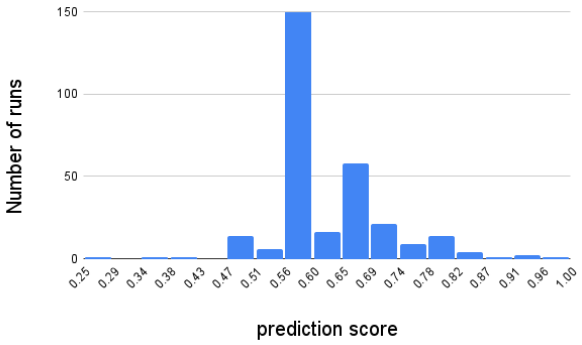

Fig. 2: Model predictions on various runs.

multiple divergence metrics between two explanations for the same example.

II. Shapley Value Based Explainers

In this section we discuss the basics of Shapley value based explainers to set the stage for the rest of the paper. Let F denote the set of features in our feature space. We use Ω_F to represent the set of all possible examples from the feature space F. Our machine learning model is a binary classifier that maps from examples to the probability of a "yes" label. Put concretely it can represented as $f_F : \Omega_F \rightarrow [0, 1]$. We use $f_F(x_F)$ to represent the model f_F being applied on a specific example $x_F \in \Omega_F$ to produce the output probability $f_F(x_F)$. The base input-independent expected value of the model is the average output of the model over all possible inputs in Ω_F i.e. $\mathbb{E}_{x_F \in \Omega_F}[f_F(x_F)]$. Note that we assume uniform distribution in computing the expectation unless specified otherwise in this paper.

In a continuation of this notation when $S \subseteq F$ is a subset of features, Ω_S is the set of all possible examples in this partial feature space. Then, $f_S : \Omega_S \rightarrow [0, 1]$ is a machine learning model that maps examples in Ω_S to the probability of a "yes" label. Then, for $x_s \in \Omega_S$, $f_S(x_S)$ is this model being applied to an example from this partial feature space.

For a subset of features S, we denote the set of missing features in F as $M = F - S$.

Given a single example $x_F \in \Omega_F$, we wish to explain our model's output, $f_F(x_F)$. The Shapley value quantifies the contribution of a feature $i \in F$ and is given by

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)].$$

The Shapley values additively explain how to get from the base input-independent expected value to the model output for x_F [12].

$$\sum_{i=1}^{|F|} \phi_i = f_F(x_F) - \mathbb{E}_{y_F \in \Omega_F}[f_F(y_F)]. \quad (1)$$

The Shapley value can therefore be interpreted as the contribution of the feature in this particular example to the deviation of the model from its expected output. Note that the definition and interpretation of Shapley values is model agnostic.

For a large number of features enumerating over all subsets can be prohibitively expensive. Therefore, it is common to instead cast the problem as constrained linear regression problem using Lasso, that fits the shapley values as weights. This approach is formalized in KernelSHAP [12] that in turn leverages LIME [11] as a subroutine. We also use KernelSHAP where the number of features becomes too large in the case of wide and deep models. However, we *still need to compute model outputs given partial features* $f_S(x_S)$ in such a setting, which leaves the problem of instability.

III. Handling Missing Features

We now consider the problem of estimating $f_S(x_S)$ when considering a subset of features $S \subseteq F$. Because we have a single model, all we have available is f_F which uses the *full* set of features. For convenience we assume the missing features are contained in the set M and therefore $F = S \cup M$ holds. Then, for $x_S \in \Omega_S$ and $z_M \in \Omega_M$, we use the notation (x_S, z_M) to denote an example in Ω_F with the combined

feature values of x_S and z_M. We also assume these are binary classification models (abuse vs. non-abuse) - although our results are extensible to multi-label classification.

The output of the classification model $f_S(x_S) \in [0,1]$ is computing the probability of the classification label y being of the positive class (in the two class problem), given the features. This is typical in abuse fighting use cases where we are interested in whether the entity is bad. We indicate this as $P(yes|x_S)$. We can then derive the following:

$$\begin{aligned} f_S(x_S) &= \Pr(yes|x_S) \\ &= \int_{\Omega_M} \Pr(yes, z_M|x_S) dz_M \\ &= \int_{\Omega_M} \Pr(yes|x_S, z_M) \Pr(z_M|x_S) dz_M \\ &= \int_{\Omega_M} f_F(x_S, z_M) \Pr(z_M|x_S) dz_M \end{aligned}$$

The last integral can be estimated via Monte Carlo integration by keeping x_S fixed and averaging $f_F(x_S, z_M)$ over the missing features z_M sampled from Ω_M using the distribution $\Pr(z_M|x_S)$. The estimator we use is:

$$\frac{\sum_{z_M \in \text{Samples}} f_F(x_S, z_M)}{|\text{Samples}|}$$

Since it uses the full set of features via f_F, we can thus dispense with the need to build models for subsets of features. Further, the law of large numbers assures us that increasing the number of samples leads to a more stable estimate of $f_S(x_S)$, something we verify in the next section.

The one wrinkle is to determine the sampling distribution over the missing features Ω_M. We observe immediately that *we do not need labeled samples but merely samples of features drawn from the underlying distribution*. Our deployment of the explainability service is able to draw samples directly from inference logs that can hold many millions of unlabeled (by humans) examples. A number of options then present themselves here.

Many implementations of Shap explainers [3], [21] assume that features are independent, which is not unreasonable since feature correlation is often eliminated during feature engineering (e.g. with PCA [27]). This assumption maps to $\Pr(z_M|x_S) = \Pr(z_M) = \Pi_{i \in M} \Pr(z_i)$ for determining samples for the Monte Carlo. Therefore we sample each feature independently to construct z_M regardless of x_S. We use this strategy in the paper to demonstrate that we can indeed obtain a stable estimate of $f_S(x_S)$ for some of our neural networks.

Given the large population of unlabeled data flowing through systems operating at scale, we can drop the independence assumption if needed. For instance, one can query logs for samples that match x_S or apply Gibbs sampling [28] to grow samples from Ω_M via sampling individual features given values of all the others. We can therefore leverage the plentiful availability of unlabeled data together with the ability to query big data (such as with Bigquery [22]) to obtain samples closer to the distribution $\Pr(z_M|x_S)$. Even with large scale data this still risks running into the curse of dimensionality. Another direction of future work for us is to exploit the observation that for fixed x_S, $\Pr(z_M|x_S) \propto \Pr(z_M|x_S)\Pr(x_S) = \Pr(x_S, z_M)$. This allows us to build a density estimator for $\Pr(x_S, z_M) = \Pr(x_F)$ on our unlabeled data (eg. [23]) and then use Metropolis-Hastings [29] to sample for $Pr(z_M|x_S)$.

IV. Evaluation

We evaluate a Shapley explainer on an abuse fighting model focusing on the stability problem. For this paper, we have used a small-scale deep neural network model with under a hundred features and under 10 layers to enable more detailed ablation studies. The dataset for this model is just under half a million examples. We also sample from this dataset, when required, for background values.

We wish to evaluate whether Monte-Carlo integration as described in Section III, indeed stabilizes both the model output when considering a subset of features as well as explanations as a whole. The key quantity we vary is the number of samples used to estimate $f_S(x_S)$. We assume independence between the features and sample from all data without regards to availability of a label.

First, we consider the problem of varying model output when changing a single feature value. As Figure 2, has already shown the results can vary dramatically from sample to sample. However, our analysis in Section III suggests that averaging the model output over some number of samples should lead to a stable estimate of the model output with the feature missing. We therefore consider the estimator $\frac{1}{N}\sum_{i=1}^{N} f_F(x_F)$ where x_F are sampled with all but the feature in question fixed. The question is whether this estimate is more stable with larger values of N i.e. a larger number of samples?

To answer this, we computed it $T = 100$ times, measuring the average and standard deviation of the estimates. Figure 3, shows the average and the companion Figure 4, shows the standard deviation. It demonstrates that the estimate indeed stabilized over a large number of samples. For this particular feature approximately 60 samples turns out to be sufficient, which is the case for most of our features.

As it is shown in the figures 5 and 6 the stability holds even when estimating model output with multiple missing features.

Next, we consider whether the explanations themselves become more stable when using a larger number of samples. To test this, we generate explanations parameterized by sample count N. For any instance where the explainer desires to compute $f_S(x_S)$, we instead use the estimate $\frac{1}{N}\sum_{i=1}^{N} f_F(x_F)$ where $x_F = x_S \cup x_M$ and each feature in x_M is sampled independently from the population of examples. We wish to show that the *explanations* are more stable with increasing N. To do this, we run $T = 100$ trials for each choice of N. Each trial generates an explanation. We compute an average "distance" between all pairs of explanations.

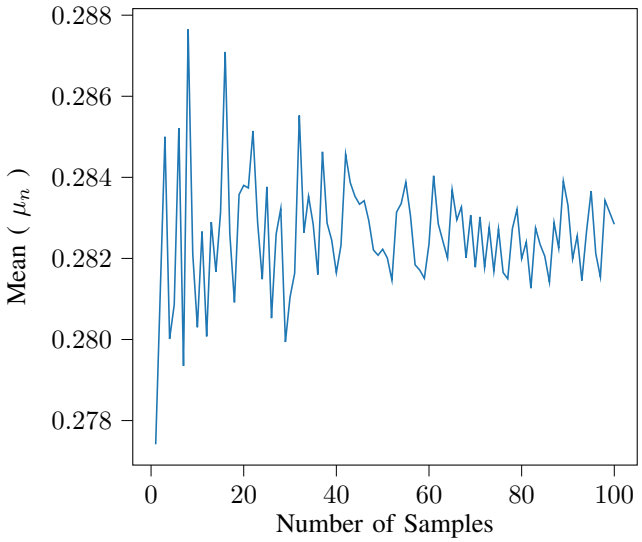

Fig. 3: Model Output vs Num Samples (single feature)

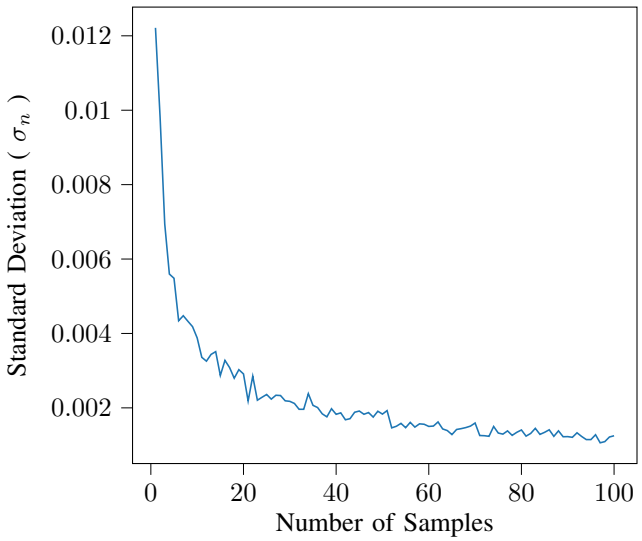

Fig. 4: Model Output vs Num Samples (single feature)

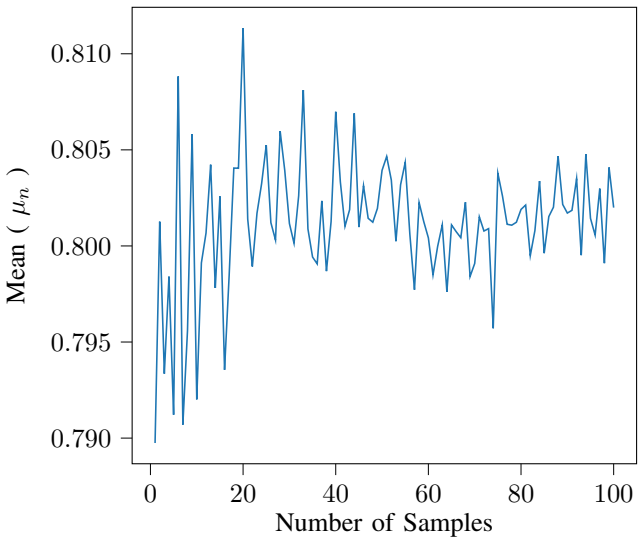

Fig. 5: Model Output vs Num Samples (multiple features)

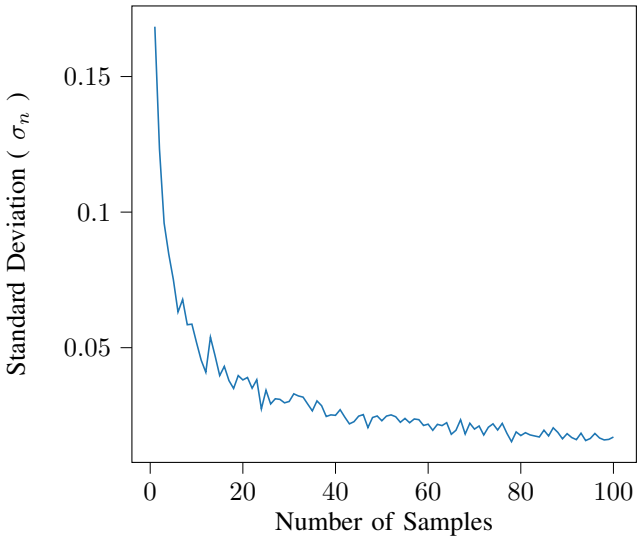

Fig. 6: Model Output vs Num Samples (multiple features)

We use three notions of distance that capture both the ranking of features as well as their individual contributions. This is because we find that most human analysts tend to focus on rankings between features in terms of importance, while ML engineers tend to also consider absolute weights and direction of contribution (Figure 2) when analyzing model performance for individual examples.

L2 Distance: We consider the shapley values generated for the features in a chosen order as a vector. We then take the L^2 norm of the difference between two vectors (i.e. the Euclidean distance) as the distance between the explanations.

JS-Divergence: Since the Shapley values must sum to the same quantity (per Equation 1) we can normalize the vector of shapley values to sum to one uniformly across all explanations. We can then treat them as probability distributions and measure divergence between them.

Let Δ_d denote the d-dimensional probability simplex:

$$\Delta_d = \{(x_1, \ldots, x_d) : x_i \geq 0, \sum_{i=1}^{d} x_i = 1\}.$$

Let $p, q \in \Delta_d$. Note, p and q define two discrete probability distributions. Then, the Kullback-Leibler divergence between p and q, denoted $D_{KL}(p, q)$ is defined as

$$D_{KL}(p, q) = \sum_{i=1}^{d} p_i \log \left(\frac{p_i}{q_i} \right).$$

This notion of divergence between distributions is not symmetric and tends to suffer from computational problems when p_i or q_i is close to zero, thus we use the following instead.

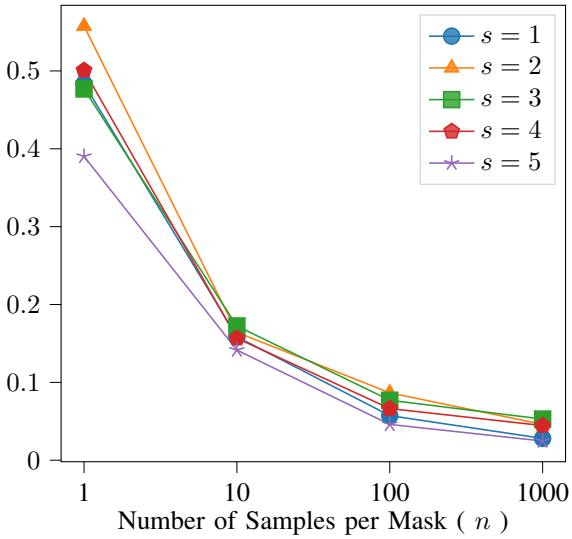

Fig. 7: Stability: L2 Distance

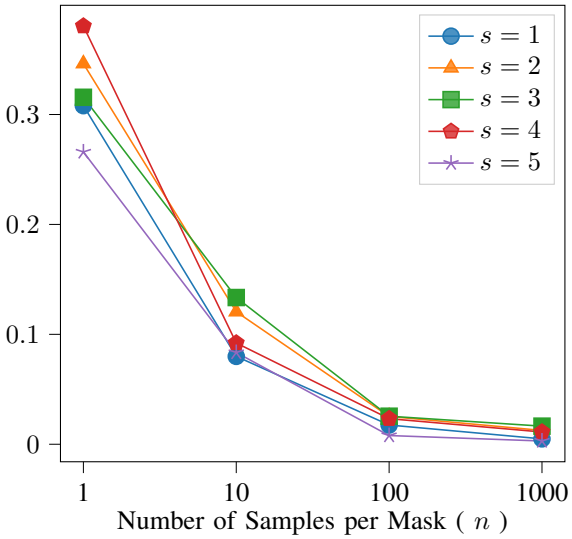

Fig. 8: Stability: Jensen-Shannon Divergence

Let $p, q \in \Delta_d$. Set $m := \frac{1}{2}(p+q) \in \Delta_d$. Again, p, q and m each define discrete probability distributions. Then, the Jensen-Shannon divergence between p and q, denoted $D_{JS}(p, q)$, is defined as

$$D_{JS}(p, q) = \frac{1}{2} D_{KL}(p, m) + \frac{1}{2} D_{KL}(q, m)$$

Spearman's rank correlation coefficient: Let $x, y \in \mathbb{R}^d$ represent two explanations (such as in Figure 1). We order the n features in x from greatest to least absolute shapley value. We can construct a vector $rg_x \in \mathbb{Z}_+^d$ by replacing each element of x by that element's rank in our ordering. We construct $rg_y \in \mathbb{Z}_+^d$ by the same process applied to y. Then, Spearman's rank correlation coefficient is given by

$$r_s(x, y) = 1 - \frac{6 \sum_{i=1}^{n} [rg_x(i) - rg_y(i)]^2}{n(n^2 - 1)},$$

Figures 7, 8 and 9 show that for all the three metrics the stability of the explanations increases, evidenced by the decrease in average distance between them, as we increase the number of samples we use. We report the results for 5 randomly chosen examples in our data ($s = 1, 2, \ldots, 5$).

This demonstrates that estimating model outputs for subsets of features via Monte Carlo integration works end-to-end in our use case thereby solving the problem of instability and unlocking the path to practical deployment.

The number of samples required to reach stable estimates of model output for a subset of features, and therefore stable explanations as a whole, varies from model to model. Although they need to be determined separately for each model, the number is small enough for practical deployment.

V. Practical Deployment

The primary uses of explainability algorithms in our machine learning workflows can be divided into two categories.

- During model development - To help model builders:
  - Feature engineering - Understand the effect of input features on the output by analyzing the contribution of a particular feature to the ML score. Explainability allows the model engineers to fine tune modeling by reasoning about whether those features truly indicate badness or are simply an artifact of the sampling process.
  - Training/Testing Data cleanup - Explainability can identify noise, bias or leakage in the training data. For example a feature with uniformly high contribution to scores across several examples can be an indication of data leakage.
- During model serving - To help model users:
  - Report the model reasoning to humans - Explainability is used as a decision aid by providing the reasoning behind an ML decision for a particular

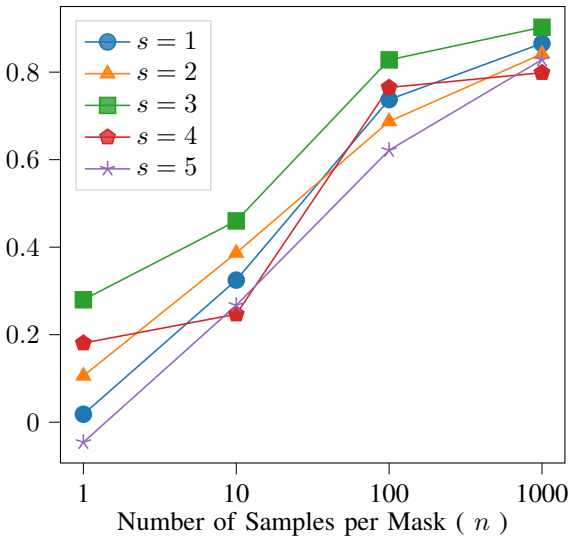

Fig. 9: Stability: Spearman's Rank Correlation Coefficient

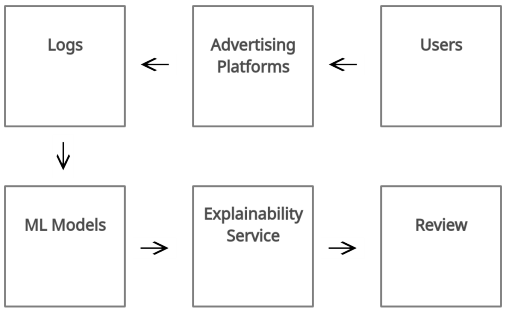

Fig. 10: Deployment of Explainability Service

prediction.

– Global analysis of groups of examples. As described in LIME [11], a submodularity analysis of the feature contributions of explanations for a group of examples can reveal surprising artifacts in how the features are used by the model. We use this to analyse large batches of examples from the model to try and discern problematic features. In some cases, this can reveal upstream issues such as bugs in features stores [20].

Explainability is available in our machine learning workflows in two forms:

- Python notebook environment: We provide a notebook that enables users to load a machine learning model checkpoint, pull features from a feature store and run the explainability algorithm to produce an explanation. This is the primary usage mode today and was used for the experiments in this paper. It is slow, due to the number of generated samples that need to be run through the model for each explanation.
- Explainability as a service - This is a service that keeps pre-loaded models in memory. At the backend, it communicates with the feature store and samples from the population of unlabeled data available in inference logs using query services. It is being built to run at scale. *Notably, generating and inferring samples around an entity of interest is trivially parallelizable making it easy to farm out the work across jobs and keep execution time per example at reasonable levels*. This is especially important as we need approximately 100 samples for a stable estimate at each invocation of $f(x_S)$.

Figure 10, depicts typical data processing pipelines that would leverage the explainability service. As part of normal operations, we regularly run inference on various entities to classify them. The explainability service samples from the logs and automatically adds an explanation note to predictions made by the models.

VI. Related work

Machine learning has been widely used in digital advertising. Examples include its usage in contextual advertising, sponsored search advertising [32] and display advertising [31]. Also several ad-targeting models such as state-of-the-art ad-targeting methods range from logistic regression [31], to log-linear models [30]. Further, complex and sophisticated modeling techniques such as factorization machines [33], deep neural networks, generative adversarial networks [34] and attention based mechanisms [35] have also been utilized in digital advertisement industry.

With the advancement and wide-scale usage of complex and non-linear modes, explainable machine learning has become an important area of research. Existing solutions to the explanation problem require the usage of interpretable models such as decision trees [8], additive models [9] and sparse linear models [10].

However, with the usage of sophisticated non-linear models, several research has been performed towards explainable machine learning. In particular, ElShawi et al. [7] presented a comprehensive experimental evaluation of several recent and popular local model agnostic inter-pretability techniques, such as, LIME [11] which aims to interpret individual model predictions based on locally approximating the model around a given prediction and SHAP [12] which assigns each feature an importance value for a particular prediction. They build upon a new class of additive feature importance measures along with theoretical results showing there is a unique solution in this class with a set of desirable properties.

Research has also been performed in summarizing existing work on explainable machine learning [15], [18], [19]. In particular, Došilović et al. [14] summarized recent developments on explainability supervised algorithms, where in they start a discussion on its connection with artificial general intelligence, and gives proposals for further research directions. Further, Vu et al. [6] introduced the c-Eval metric and the corresponding framework to quantify the explainer's quality on feature-based explainers of machine learning image classifiers. Given a prediction and the corresponding explanation on that prediction, c-Eval is the minimum-power perturbation that successfully alters the prediction while keeping the explanation's features unchanged.

Explainable machine learning has also been studied across various application areas. In particular, Lundberg et al. [13] worked on the of explainable machine-learning predictions for the prevention of hypoxaemia during surgery. Zitnik et al. [16] worked on interpretable machine learning for integrating data in biology and medicine. Ghosal et al. [17] extended the concept of explainable deep machine learning to automate the process of plant stress identification, classification, and quantification.

However, to the best of our knowledge this is the first manuscript related to practical deployment in conjunction with the stability of the explainable machine learning classifiers at scale in the digital advertisement industry.

VII. Conclusion

Efficient digital advertising platforms aim to map customer advertisements to a targeted set of audience as this set has a

high propensity to respond positively towards the advertisements. In order to achieve this successful mapping, several machine learning models are utilized. Further, advertising platforms are often susceptible to abuse stemming from bad actors aiming to serve malicious and undesirable content to viewers of advertisements.

Detection and deterrence of such abuse then requires detailed understanding of complex machine learning model decisions along with establishing a clear link from features to model output. The importance of understanding model output usually stems from a human reviewer's ability to understand existing model decisions.

Shapely values are an intuitive way to understand model output for human interpretation. The key challenge in computing Shapley values usually stems from its inability to compute the model output for subsets of features, as training a new model for every subset would result in a number of models exponential in the number of features.

In order to overcome the aforementioned challenge, in this manuscript we have shown a simple and explicit link between estimating a model's outputs given a subset of features and Monte-Carlo integration, in a model agnostic way. To the best of our knowledge, such an explicit study in an applied setting has not been discussed in terms of estimating model output given missing features. We also present the reduction in variance in the estimate of a model's output given subsets of missing features for our models. Lastly, we demonstrate how explanations become more stable (e.g. by measuring multiple divergence metrics) across explanations for the same example. We also discuss how this unlocks the application of Shapley value based explainers in our machine learning models.

VIII. Acknowledgment

We would like to thank the anonymous reviewers for their detailed feedback that greatly contributed to the final quality of the exposition. We would also like to thank our partner analysts who took the time to work with us in guiding the use of explainability for machine learning models towards useful deployment.

References

[1] https://ads.google.com

[2] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[3] Lundberg, Scott M., and Su-In Lee."A unified approach to interpreting model predictions." In Proceedings of the 31st international conference on neural information processing systems, pp. 4768-4777. 2017.

[4] Hershey, John R., and Peder A. Olsen. "Approximating the Kullback Leibler divergence between Gaussian mixture models." In 2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07, vol. 4, pp. IV-317. IEEE, 2007.

[5] Fuglede, Bent, and Flemming Topsoe. "Jensen-Shannon divergence and Hilbert space embedding." In International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings., p. 31. IEEE, 2004.

[6] Vu, Minh N., Truc D. Nguyen, NhatHai Phan, Ralucca Gera, and My T. Thai. "Evaluating explainers via perturbation." (2019).

[7] ElShawi, Radwa, Youssef Sherif, Mouaz Al-Mallah, and Sherif Sakr. "Interpretability in healthcare: A comparative study of local machine learning interpretability techniques." Computational Intelligence (2020).

[8] Wang, Fulton, and Cynthia Rudin. "Falling rule lists." In Artificial Intelligence and Statistics, pp. 1013-1022. PMLR, 2015.

[9] Caruana, Rich, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission." In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 1721-1730. 2015.

[10] Ustun, Berk, and Cynthia Rudin. "Supersparse linear integer models for optimized medical scoring systems." Machine Learning 102, no. 3 (2016): 349-391.

[11] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "" Why should i trust you?" Explaining the predictions of any classifier." In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 1135-1144. 2016.

[12] Lundberg, Scott M., and Su-In Lee. "A unified approach to interpreting model predictions." In Proceedings of the 31st international conference on neural information processing systems, pp. 4768-4777. 2017.

[13] Lundberg, Scott M., Bala Nair, Monica S. Vavilala, Mayumi Horibe, Michael J. Eisses, Trevor Adams, David E. Liston et al. "Explainable machine-learning predictions for the prevention of hypoxaemia during surgery." Nature biomedical engineering 2, no. 10 (2018): 749-760.

[14] Došilović, Filip Karlo, Mario Brčić, and Nikica Hlupić. "Explainable artificial intelligence: A survey." In 2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO), pp. 0210-0215. IEEE, 2018.

[15] Carvalho, Diogo V., Eduardo M. Pereira, and Jaime S. Cardoso. "Machine learning interpretability: A survey on methods and metrics." Electronics 8, no. 8 (2019): 832.

[16] Zitnik, Marinka, Francis Nguyen, Bo Wang, Jure Leskovec, Anna Goldenberg, and Michael M. Hoffman. "Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities." Information Fusion 50 (2019): 71-91.

[17] Ghosal, Sambuddha, David Blystone, Asheesh K. Singh, Baskar Ganapathysubramanian, Arti Singh, and Soumik Sarkar. "An explainable deep machine vision framework for plant stress phenotyping." Proceedings of the National Academy of Sciences 115, no. 18 (2018): 4613-4618.

[18] Arrieta, Alejandro Barredo, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI." Information Fusion 58 (2020): 82-115.

[19] Adadi, Amina, and Mohammed Berrada. "Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)." IEEE access 6 (2018): 52138-52160.

[20] https://cloud.google.com/blog/products/ai-machine-learning/introducing-feast-an-open-source-feature-store-for-machine-learning

[21] Štrumbelj, E., Kononenko, I. "Explaining prediction models and individual predictions with feature contributions." Knowl Inf Syst 41, 647–665 (2014).

[22] https://cloud.google.com/bigquery

[23] Mathieu Germain, Karol Gregor, Iain Murray, Hugo Larochelle. "MADE: Masked Autoencoder for Distribution Estimation." Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:881-889, 2015.

[24] https://github.com/slundberg/shap

[25] Winter, Eyal. "The shapley value." Handbook of game theory with economic applications 3 (2002): 2025-2054.

[26] Kong, A., P. McCullagh, X-L. Meng, D. Nicolae, and Z. Tan. "A theory of statistical models for Monte Carlo integration." Journal of the Royal Statistical Society: Series B (Statistical Methodology) 65, no. 3 (2003): 585-604.

[27] Wold, Svante, Kim Esbensen, and Paul Geladi. "Principal component analysis." Chemometrics and intelligent laboratory systems 2, no. 1-3 (1987): 37-52.

[28] Gelfand, Alan E. "Gibbs sampling." Journal of the American statistical Association 95, no. 452 (2000): 1300-1304.

[29] Chib, Siddhartha, and Edward Greenberg. "Understanding the metropolis-hastings algorithm." The american statistician 49, no. 4 (1995): 327-335.

[30] Agarwal, Deepak, Rahul Agrawal, Rajiv Khanna, and Nagaraj Kota. "Estimating rates of rare events with multiple hierarchies through scalable log-linear models." In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 213-222. 2010.

[31] Chapelle, Olivier, Eren Manavoglu, and Romer Rosales. "Simple and scalable response prediction for display advertising." ACM Transactions on Intelligent Systems and Technology (TIST) 5, no. 4 (2014): 1-34.
[32] Fain, Daniel C., and Jan O. Pedersen. "Sponsored search: A brief history." Bulletin of the american Society for Information Science and technology 32, no. 2 (2006): 12-13.
[33] Juan, Yuchin, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. "Field-aware factorization machines for CTR prediction." In Proceedings of the 10th ACM conference on recommender systems, pp. 43-50. 2016.
[34] Doan, Khoa D., Pranjul Yadav, and Chandan K. Reddy. "Adversarial factorization autoencoder for look-alike modeling." In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 2803-2812. 2019.
[35] Liu, Yudan, Kaikai Ge, Xu Zhang, and Leyu Lin. "Real-time Attention Based Look-alike Model for Recommender System." In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2765-2773. 2019.