Journal of Machine Learning Research 24 (2023) 1-38 Submitted 5/22; Revised 8/23; Published 9/23

Deletion and Insertion Tests in Regression Models

Naofumi Hama NAOFUMI.HAMA.HD@HITACHI.COM
Hitachi, Ltd.
Research & Development Group
Kokubunji, Tokyo, 185-8601, Japan

Masayoshi Mase MASAYOSHI.MASE.MH@HITACHI.COM
Hitachi, Ltd.
Research & Development Group
Kokubunji, Tokyo, 185-8601, Japan

Art B. Owen OWEN@STANFORD.EDU
Department of Statistics
Stanford University
Stanford, CA 94305, USA

Editor: Francis Bach

Abstract

A basic task in explainable AI (XAI) is to identify the most important features behind a prediction made by a black box function f. The insertion and deletion tests of Petsiuk et al. (2018) can be used to judge the quality of algorithms that rank pixels from most to least important for a classification. Motivated by regression problems we establish a formula for their area under the curve (AUC) criteria in terms of certain main effects and interactions in an anchored decomposition of f. We find an expression for the expected value of the AUC under a random ordering of inputs to f and propose an alternative area above a straight line for the regression setting. We use this criterion to compare feature importances computed by integrated gradients (IG) to those computed by Kernel SHAP (KS) as well as LIME, DeepLIFT, vanilla gradient and input$\times$gradient methods. KS has the best overall performance in two datasets we consider but it is very expensive to compute. We find that IG is nearly as good as KS while being much faster. Our comparison problems include some binary inputs that pose a challenge to IG because it must use values between the possible variable levels and so we consider ways to handle binary variables in IG. We show that sorting variables by their Shapley value does not necessarily give the optimal ordering for an insertion-deletion test. It will however do that for monotone functions of additive models, such as logistic regression.

Keywords: Aumann-Shapley value, Deletion test, Explainable AI, Insertion test, Integrated gradients, Shapley value

1. Introduction

Explainable AI methods are used help humans learn from patterns that a machine learning or artificial intelligence model has found, or to judge whether those patterns are scientifically reasonable or whether they treat subjects fairly. As Hooker et al. (2019) note, there is no ground truth for explanations. Mase et al. (2022) attribute this to the greater difficulty of identifying causes of effects compared to effects of causes (Dawid and Musio, 2021).

Lacking a ground truth, researchers turn to axioms and sanity checks to motivate and vet explanatory methods. There are also some numerical measures that one can use to compute a quality measure for methods that rank variables from most to least important. These include the Area Over Perturbation Curve (AOPC) of Samek et al. (2016) and the Area Under the Curve (AUC) measure of Petsiuk et al. (2018) that we focus on. They have the potential to augment intuitive and philosophical distinctions among methods with precise numerical comparisons. In this paper we make a careful study of the properties of those measures and we illustrate their use on two datasets.

Insertion and deletion tests were used by Petsiuk et al. (2018) to compare variable importance methods for black box functions. In their specific case they had an image classifier that would, for example, conclude with high confidence that a given image contains a mountain bike. Then the question of interest was to identify which pixels are most important to that decision. They propose to delete pixels, replacing them by a plain default value (constant values such as black or the average of all pixels from many images) in order from most to least important for the decision that the image was of the given class. If they have ordered the pixels well, then the confidence level of the predicted class should drop quickly as more pixels are deleted. By that measure, their Randomized Input Sampling for Explanation (RISE) performed well compared to alternatives such as GradCAM (Selvaraju et al., 2017) and LIME (Ribeiro et al., 2016) when explaining outputs of a ResNet50 classifier (Zhang et al., 2018). For instance, Figure 2 of Petsiuk et al. (2018) has an example where occlusion of about 4% of pixels, as sorted by their RISE criterion can overturn the classification of an image. They also considered an insertion test starting from a blurred image of the original one and inserting pixels from the real image in order from most to least important. An ordering where the confidence rises most quickly is then to be preferred. Petsiuk et al. (2018) scored their methods by an area under the curve (AUC) metric that we will describe in detail below. The idea to change features in order of importance and score how quickly predictions change is quite natural, and we expect many others have used it. We believe that our analysis of the methods in Petsiuk et al. (2018) will shed light on other similar proposals.

Figure 1 shows an example where an image is correctly and confidently classified as an albatross by an algorithm described in Appendix A.1. The integrated gradients (IG) method of Sundararajan et al. (2017) that we define below can be used to rank the pixels by importance. In this instance the deletion AUC is 0.27 which can be interpreted as meaning that about 27% of the pixels have to be deleted before the algorithm completely forgets that the image is of an albatross. The model we used accepts square images of size 224×224 pixels with 3 color channels. As a preprocessing step, we cropped the leftmost image in Figure 1 to its central 224×224 pixels. The IG feature attributions from each pixel (summed over red, green and blue channels) of this square image are presented in the rightmost panel of Figure 1. A saliency map shows that pixels in the bird's face, especially the eye and beak are rated as most important.

In this paper we study insertion and deletion metrics for uses that include regression problems in addition to classification. We consider a function $f(\boldsymbol{x})$ such as an estimate of a response y given n predictors represented as components of $\boldsymbol{x}$. The regression context is different from classification. The trajectory taken by $f(\cdot)$ as inputs are switched one by one from a point $\boldsymbol{x}$ to a baseline point $\boldsymbol{x}'$ can be much less monotone than in the image

classification problems that motivated Petsiuk et al. (2018). We don't generally find that either $f(\boldsymbol{x})$ or $f(\boldsymbol{x}')$ is near zero. There are also use cases where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are both actual observations; it is not necessary for one of them to be an analogue of a completely gray image or otherwise null data value. Sometimes $f(\boldsymbol{x}) \approx f(\boldsymbol{x}')$ and yet it can still be interesting to understand what happens to f when some components of $\boldsymbol{x}$ are changed to corresponding values of $\boldsymbol{x}'$.

Our main contributions are as follows. Despite these differences between classification and regression, we find that insertion and deletion metrics can be naturally extended to regression problems. We then develop expressions for the resulting AUC in terms of certain main effects and interactions in f building on the anchored decomposition from Kuo et al. (2010) and others. This anchored decomposition is a counterpart to the better known analysis of variance (ANOVA) decomposition. The anchored decomposition does not require a distribution on its inputs, much less the independence of those inputs that the ANOVA requires. In the regression context we prefer to change the AUC computation replacing the horizontal axis by a straight line connecting $f(\boldsymbol{x})$ to $f(\boldsymbol{x}')$. We obtain an expression for the average AUC in a case where variables were inserted in a uniform random order over all possible permutations. In settings without interactions the area between the variable change curve (that we define below) and the straight line has expected value zero under those permutations, but interactions change this. We also show that the expected area between the insertion curve and an analogous deletion curve that we define below does have expected value zero, even in the presence of interactions of any order. Some other contributions described below show that in some widely used models the same ordering that optimizes an area criterion also optimizes a Shapley value.

We take a special interest in the integrated gradients (IG) method of Sundararajan et al. (2017) because it is very fast. The number of function or derivative evaluations that it requires grows only linearly in the number of input variables, for any fixed number of evaluation nodes in the Riemann sum it uses to approximate an integral. The cost of exact computation for kernel SHAP (KS) of Lundberg and Lee (2017) grows exponentially with the number of variables, although it can be approximated by sampling. We also include LIME of Ribeiro et al. (2016), DeepLIFT of Shrikumar et al. (2017), Vanilla Grad of Simonyan et al. (2013) and input times gradient method of Shrikumar et al. (2016). In the datasets we considered, KS is generally best overall. We note that the term 'Vanilla' is not used by Simonyan et al. (2013) to describe their methods but it has been used by others, such as Agarwal et al. (2022).

It is very common for machine learning functions to include binary inputs. For this reason we discuss how to extend IG to handle some dichotomous variables and then compare it to the other methods, especially KS. Simply extending the domain of f for such variables from $\{0, 1\}$ to $[0, 1]$ is easy to do and it avoids the exponential cost that some more principled choices have.

The remainder of this paper is organized as following. Section 2 cites related works, introduces some notation and places our paper in the context of explainable AI (XAI), while also citing some works that express misgivings about XAI. Section 3 defines the AUC and gives an expression for it in terms of main effects and interactions derived from an anchored decomposition of the prediction function f. The expected AUC is obtained for a random ordering of variables. We also introduce an area between the curves (ABC)

quantity using a linear interpolation baseline curve instead of the horizontal axis. We show that arranging input variables in decreasing order by Shapley value does not necessarily give the order that maximizes AUC, due to the presence of interactions. Models that, like logistic regression are represented by an increasing function of an additive model do get their greatest AUC from the Shapley ordering depsite the interactions introduced by the increasing function. When that function is differentiable, then IG finds the optimal order. Section 4 discusses how to extend IG to some dichotomous variables. We consider three schemes: simply treating binary variables in $\{0, 1\}$ as if they were continuous values in $[0, 1]$, multilinear interpolation of the function values at binary points, and using paths that jump from $x_j = 0$ to $x_j = 1$. The simple strategy of casting the binary inputs to $[0, 1]$, which many prediction functions can do, is preferable on grounds of speed. Section 5 presents some empirical work. We choose a regression problem about explaining the value of a house in Bangalore using some data from Kaggle (Section 5.1). This is a challenging prediction problem because the values are heavily skewed. It is especially challenging for IG because all but two of the input variables are binary and IG is defined for continuous variables. The model we use is a multilayer perceptron. We compare variable rankings from six methods. KS is overall best but IG is much faster and nearly as good. Section 5.2 looks at a problem from high energy physics using data from CERN. Section 6 includes a ROAR analysis that compares how well the KS and IG measures rank the importance of variables in a setting where the model is to be retrained without its most important variables. Section 7 has some final comments. Appendix A gives some details of the data and models we study. Theorem 3 is proved in Appendix B.

2. Related Work

The insertion and deletion measures we study are part of XAI. Methods from machine learning and artificial intelligence are being deployed in electronic commerce, finance and other industries. Some of those applications are mission critical (e.g., in medicine, security or autonomous driving). When models are selected based on their accuracy on holdout sets, the winning algorithms can be very complicated. There is then a strong need for human understanding of how the methods work in order to reason about their accuracy on future data. For discussions on the motivations and methods for XAI see recent surveys such as Liao and Varshney (2021), Saeed and Omlin (2023) and Bodria et al. (2023).

One of the most prominent XAI tasks is attribution in which one quantifies and compares importance of the model inputs to the resulting prediction. Since there is no ground-truth for explanations, these attributions are usually compared based on theoretical justifications. From this viewpoint, SHAP (SHapley Additive exPlanations) (Lundberg and Lee, 2017) and IG (Sundararajan et al., 2017) are popular feature importance methods due to their grounding in cooperative game theory. Given some reasonable axioms, game theory can produce unique variable importance measures in terms of Shapley values and Aumann-Shapley values respectively.

As a complementary method to theoretical a priori justification, one can also examine the outputs of attribution methods numerically, either applying sanity checks or computing quantitative quality measures. The insertion and deletion tests of Petsiuk et al. (2018) that we study are of this type. Quantitative metrics for XAI were recently surveyed in Nauta

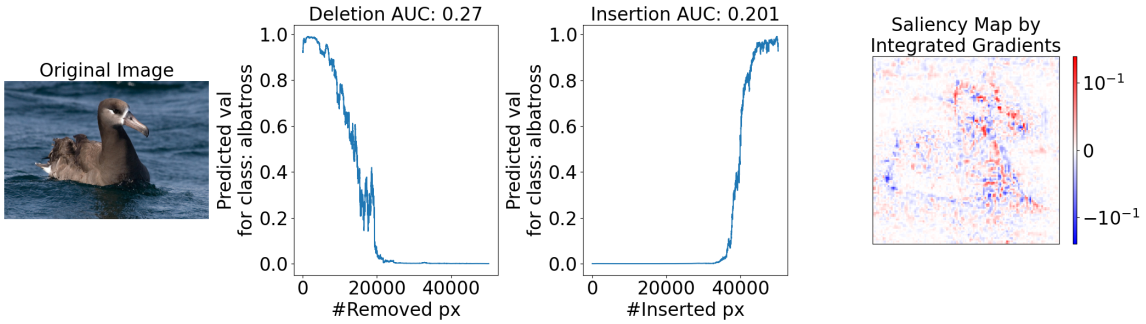

Figure 1: An example of deletion and insertion tests in image classification for an image from Wah et al. (2011).

et al. (2022) who also discuss metrics based on whether the importance ratings align with those of human judges.

As an example of sanity checks, Adebayo et al. (2018) find that some algorithms to produce saliency maps are nearly unchanged when one scrambles the category labels on which the algorithm was trained, or when one replaces trained weights in a neural network by random weights. They also find that some saliency maps are very close to what edge detectors would produce and therefore do not make much use of the predicted or actual class of an image. Another example from Adebayo et al. (2020) checks whether saliency methods can detect spurious correlation in a setting where the backgrounds in training images are artificially correlated to output labels and the model learns this spurious correlation correctly.

One important method is the ROAR (RemOve And Retrain) approach of Hooker et al. (2019). It sequentially removes information from the least important columns in each observation in the training data and retrains the model with these partially removed training data iteratively. A good variable ranking will show rapidly decreasing performance of the classifier as the most important inputs are removed. The obvious downside of this method is that it requires a lot of expensive retraining that insertion and deletion methods avoid.

The insertion and deletion tests we study have been criticized by Gomez et al. (2022) who note that the synthesized images generated in these tests are unnatural and do not resemble the images on which the algorithms were trained. The same issue of unnatural inputs was raised by Mase et al. (2019). Gomez et al. (2022) also point out that insertion and deletion tests only compare the rankings of the inputs.

Fel et al. (2021) noted that scores on insertion tests can be strongly influenced by the first few pixels inserted into a background image. They then note that images with only a few non-background pixels are quite different from the target image. The policy choice of what baseline to compare an image to affects whether the result can be manipulated. An adversarily selected image might differ from a target image in only a few pixels, and yet have a very different classification. At the same time both of these images differ greatly from a neutral background image. An insertion test comparing the target image to a blurred background might show that the classification depends on many pixels, while a deletion

test with the adversarial image will show that only a few pixels need to change. The two tests will thus disagree on whether the classification was influenced by few or many pixels. Both are correct because they address different issues.

A crucial choice in using insertion and deletion tests is which reference input to use when deleting or inserting variables. Petsiuk et al. (2018) decided to insert real pixels into a blurred image instead of inserting them into a solid gray image because, when the most important pixels form an ellipsoidal shape, then inserting them into a gray background might lead a spurious classification (such as 'balloon'). On the other hand, deleting pixels via blurring might underestimate the salience of those pixels if the algorithm is good at inferring from blurred images. In our albatross example of Figure 1 we used an all black image with $224 \times 224 \times 3$ zeros. We call such choices 'policies' and note that a good policy choice depends on the scientific goals of the study and the strengths and weaknesses of the algorithm under study.

Haug et al. (2021) study different kinds of baseline images for image classification problems noting that the choice of background affects how well a method performs. Among the backgrounds they mention are constants, blurred images, uniform or Gaussian noise, average images and baseline images maximally distant from the target image. They also mention the neutral backgrounds of Izzo et al. (2020) that lie on a decision boundary. Sundararajan and Najmi (2020) discuss taking every available data point as a baseline and averaging the resulting Shapley values. Sturmfels et al. (2020) note the importance of choosing baselines carefully, pointing out that zero would be a bad baseline for blood sugar and that if a solid color image is used as a baseline in image classification, then the result cannot attribute importance to pixels of that color. They discuss many of the baselines that Haug et al. (2021) do and they propose the 'farthest image' baseline.

There are also broader criticisms of XAI methods. Kumar et al. (2020) point out some difficulties in formulating a variable importance problem as a game suitable for use in Shapley value. Their view is that those explanations do not match what people expect from an explanation. They also identify a catch-22 where either allowing or disallowing a variable not used by a model to be important causes difficulties. Rudin (2019) says that one should not use a black box model to make high stakes decisions but should instead only use interpretable models. She also disputes that this would necessarily cause a loss in accuracy. We see a lot of value in that view, and we know that there are examples where interpretable models perform essentially as well as the best black boxes. However black boxes are very widely used. There is then value in XAI methods that can reveal and quantify any of their flaws. Furthermore, an explanation depends on not just the form of the model but also on the joint distribution of the predictor variables. For example an interpretable model that does not use the race of a subject might still have a discriminatory impact due to associations among the predictors and XAI methods can be used to evaluate such bias.

Prior uses of insertion and deletion tests have mostly been about image or text classification. There have been a few papers using them for tabular data or time series data, such as Cai et al. (2021), Hsieh et al. (2021), Parvatharaju et al. (2021), and Ismail et al. (2020).

Ancona et al. (2017) also describe a strategy of changing variables one at a time and observing how the quality of a prediction changes in response independently of Petsiuk et al. (2018). Their Figure 3c compares the trajectories taken by several different methods on some image classification problems. They have insertion and deletion curves to compare an

occlusion method to integrated gradients. Unlike Petsiuk et al. (2018), they do not report an AUC quantity.

The work of Samek et al. (2016) precedes Petsiuk et al. (2018) and uses the same global ablation strategy of deleting information in order from most to least important. Their deletions involve replacing a whole block of pixels (e.g., a 9×9 block) by uniformly distributed noise. One motivation for using such noise was to generate images outside the manifold of natural images. Their average over a perturbation curve is comparable to the average under the deletion curve of Petsiuk et al. (2018). Where Petsiuk et al. (2018) focus on curves for individual images, Samek et al. (2016) study the average of such curves over many images. In their numerical work they only make the first 100 such perturbations, affecting about 1/6 of the pixels.

For us the advantage of the approach in Petsiuk et al. (2018) is that their AUC is defined by running the variable substitution to completion, changing every feature x_j to the baseline value x'_j. Then we use two orders, one that seeks to increase f as fast as possible and one that seeks to decrease it as fast as possible, and study the area between those two curves.

3. The AUC for Regression

The AUC method for comparing variable rankings has not had much theoretical analysis yet. This section develops some of its properties. The development is quite technical in places. We begin with a non technical account emphasizing intuition. Readers may use the intuitive development as orientation to the technical parts or they may prefer to skip the technical parts.

If we order the inputs to a function f and then change them one at a time from the value in point $\boldsymbol{x}$ to that in a baseline value $\boldsymbol{x}'$, the resulting function values trace out a curve over the interval $[0, n]$. If we have tried to order the variables starting with those that will most increase f and ending with those that least increase (most decrease) f then a better ordering is one that gives a larger area under the curve (AUC). We will find it useful to consider the signed area under this curve but above a straight line connecting the end points. This area between the curves (ABC) is the original AUC minus the area under the line segment.

A deletion measure orders the variables from those thought to be most decreasing of f to those that are least decreasing, i.e., the opposite order to an insertion test. For deletion we like to use the area ABC below the straight line but above the curve that the deletion process traces out. When we need to refer to insertion and deletion ABCs in the same expression we use ABC$'$ for the deletion case.

Additive functions are of special interest because additivity simplifies explanation and such models often come close to the full predictive power of a more complicated model. For an additive model, the incremental change in f from replacing x_j by x'_j, call it Δ_j, does not depend on x_k for $k \neq j$. In this case the AUC is maximized by ordering the variables so that $\Delta_1 \geqslant \Delta_2 \geqslant \cdots \geqslant \Delta_n$. In this case the Shapley values are $\phi_j = \Delta_j$ and so ordering by Shapley value maximizes both AUC and ABC. We also show that if one orders the predictors randomly, then the expected value of ABC under this randomization is zero for an additive function.

Prediction functions must also capture interactions among the input variables. We study those using some higher order differences of differences. This way of quantifying interactions comes from an anchored decomposition that we present. This anchored decomposition is a less well known alternative to the analysis of variance (ANOVA) decomposition. The interaction quantity for a set $u \subseteq \{1, 2, \dots, n\}$ of variables is denoted Δ_u. This interaction does not contribute to any points along the curve until the 'last' member of the set u, denoted $\lceil u \rceil$, has been changed. It is thus present only in the final $n + 1 - \lceil u \rceil$ points of the insertion curve. A large AUC comes not just from bringing the large main effects to the front of the list. It also helps to have all elements in a positive interaction Δ_u included early, and at least one element in a negative interaction Δ_u appear very late in the ordering.

When there are interactions present, it is no longer true that $\mathbb{E}(ABC)$ must be zero under random ordering. We show in Appendix B.1 that the area between the insertion and deletion curves $\mathrm{ABC} + \mathrm{ABC}'$ satisfies $\mathbb{E}(\mathrm{ABC} + \mathrm{ABC}') = 0$ under random ordering of inputs whether or not interactions are present.

Section 3.4 shows that if we sort variables in decreasing order of their Shapley values, then we do not necessarily get the ordering that maximizes the AUC. This is natural: the Shapley value ϕ_j of variable j is defined as a weighted sum of 2^{n-1} incremental values for changing x_j, while the AUC, uses only one of those incremental values for variable j. The anchored decomposition that we present below makes it simple to construct an example with $n = 3$ where the Shapley values are $\phi_1 > \phi_2 > \phi_3$ while the order $(1, 3, 2)$ has greater AUC than the order $(1, 2, 3)$. This is not to say that insertion AUCs are somehow in error for not being optimized by the Shapley ordering, nor that Shapley value is in error for not optimizing the AUC. The two measures have different definitions and interpretations. They can reasonably be considered proxies for each other, but the Shapley value weights a variable's interactions in a different way than the AUC does.

In Appendix B.2 we consider the logistic regression model $f(\boldsymbol{x}) = \Pr(Y = 1 \mid \boldsymbol{x}) = (1+\exp(-\beta_0 - \boldsymbol{x}^{\mathsf{T}}\beta))^{-1}$. Because of the curvature of the logistic transformation, this function has interactions of all orders. At the same time $\tilde{f}(\boldsymbol{x}) = \log(f(\boldsymbol{x})/(1 - f(\boldsymbol{x}))) = \beta_0 + \boldsymbol{x}^{\mathsf{T}}\beta$ is additive so on this scale the Shapley ordering does maximize AUC. The AUC on the original probability scale is perhaps the more interpretable choice. We show that due to the monotonicity of the logistic transformation, the Shapley ordering for $f(\boldsymbol{x})$ is the same as for $\tilde{f}(\boldsymbol{x})$ and so it also maximizes the AUC for $f(\boldsymbol{x}) = \Pr(Y = 1 \mid \boldsymbol{x})$. Because $\exp(\cdot)$ is strictly monotone the Shapley ordering also optimizes AUC for loglinear models and for naive Bayes. It is also shown there that for a differentiable increasing function of an additive function that integrated gradients will compute the optimal order.

The next subsections present the above findings in more detail. Some of the derivations are in an appendix. We use well known properties of the Shapley value. Those are discussed in many places. We can recommend the recent reference by Plischke et al. (2021) because it also discusses Harsanyi dividends and is motivated by variable importance.

3.1 ABC Notation

We study an algorithm $f : \mathcal{X} \to \mathbb{R}$ that makes a prediction based on input data $\boldsymbol{x} \in \mathcal{X} = \prod_{j=1}^{n} \mathcal{X}_j$. The points $\boldsymbol{x} \in \mathcal{X}$ are written $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$. In most applications $\mathcal{X}_j \subseteq \mathbb{R}$. While some attribution methods require real-valued features, the AUC quantity we present

does not require it. For classification, f could be the estimated probability that a data point with features $\boldsymbol{x}$ belongs to class y, or it could be that same probability prior to a softmax normalization. Our emphasis is on regression problems.

The set of variable indices is $1{:}n \equiv \{1, 2, \ldots, n\}$. For any $u \subseteq 1{:}n$ we write $\boldsymbol{x}_u$ for the components x_j that have $j \in u$. We write $-u$ for $1{:}n \setminus u$. We often need to merge indices from two or more points into one hybrid point. For this $\boldsymbol{x}_u{:}\boldsymbol{x}'_{-u}$ is the point $\tilde{\boldsymbol{x}} \in \mathcal{X}$ with $\tilde{x}_j = x_j$ for $j \in u$ and $\tilde{x}_j = x'_j$ for $j \notin u$. That is, the parts of $\boldsymbol{x}$ and $\boldsymbol{x}'$ have been properly assembled in such a way that we can pass the hybrid to f getting $f(\tilde{\boldsymbol{x}})$. More generally for disjoint u, v, w with $u \cup v \cup w = 1{:}n$ the point $\boldsymbol{x}_u{:}\boldsymbol{y}_v{:}\boldsymbol{z}_w$ has components x_j, y_j and z_j for j in u, v and w respectively.

The cardinality of u is denoted $|u|$. We also write $\lceil u \rceil = \max\{j \in 1{:}n \mid j \in u\}$ with $\lceil \varnothing \rceil = 0$ by convention. It is typographically convenient to shorten the singleton $\{j\}$ to just j where it could only represent a set and not an integer, especially within subscripts.

Suppose that we have two points $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ and are given a method to attribute the difference $f(\boldsymbol{x}') - f(\boldsymbol{x})$ to the variables $j \in 1{:}n$. This method produces attribution values $A_f(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}^n$ with the interpretation that $A_f(\boldsymbol{x}, \boldsymbol{x}')_j$ is a measure of the effect on f of changing x_j to x'_j. We can then sort the variables $j \in 1{:}n$ according to their attribution values $A_f(\boldsymbol{x}, \boldsymbol{x}')_j$. In an insertion test we insert the variables from $\boldsymbol{x}'$ into $\boldsymbol{x}$ in order from ones thought to most increase $f(\cdot)$ (i.e., largest $A_f(\boldsymbol{x}, \boldsymbol{x}')_j$) to ones thought to most decrease $f(\cdot)$ (smallest $A_f(\boldsymbol{x}, \boldsymbol{x}')_j$). Let the constructed points be $\tilde{\boldsymbol{x}}^{(j)}$ for $j = 0, 1, \ldots, n$ with $\tilde{\boldsymbol{x}}^{(0)} = \boldsymbol{x}$ and $\tilde{\boldsymbol{x}}^{(n)} = \boldsymbol{x}'$. If we have chosen a good order there will be a large area under the curve $(j, f(\tilde{\boldsymbol{x}}^{(j)}))$ for $j = 0, 1, \ldots, n$ and consequently also a large (signed) area between that curve and a straight line connecting its endpoints. The left panel in Figure 2 illustrates ABC for insertion.

In a deletion measure we order the variables from the ones thought to have the most negative effect on f to the ones thought to have the most positive effect. Those variables are changed from x_j to x'_j in that order and a good ordering creates a curve with a small area under it. We keep score by using the signed area above that curve but below the straight line connecting $f(\boldsymbol{x})$ to $f(\boldsymbol{x}')$. Note that we are still inserting variables from $\boldsymbol{x}'$ into $\boldsymbol{x}$ but, in an analogy to what happens in images we are deleting the information that we think would make f large, which in that setting made the algorithm confident about what was in the image. Let $\check{\boldsymbol{x}}^{(j)}$ be the point we get after placing the j elements of $\boldsymbol{x}'$ thought to most decrease f into $\boldsymbol{x}$. Our ABC criterion for deletion is the signed area above the curve $(j, f(\check{\boldsymbol{x}}^{(j)}))$ but below the straight line connecting connecting the endpoints. The right panel in Figure 2 illustrates ABC for deletion.

We also considered taking insertion to mean replacing components x_j by x'_j in increasing order of predicted change to f when $f(\boldsymbol{x}') > f(\boldsymbol{x})$ and taking deletion to mean replacement starting with the most negative changes when $f(\boldsymbol{x}') < f(\boldsymbol{x})$. This convention may seem like a natural extension of the uses in image classification, but it has two difficulties for regression. First, it is not well defined when $f(\boldsymbol{x}) = f(\boldsymbol{x}')$. Second, while this exact equality might seldom hold, that definition makes cases $f(\boldsymbol{x}') = f(\boldsymbol{x}) + \epsilon$ very different from those with $f(\boldsymbol{x}') = f(\boldsymbol{x}) - \epsilon$, for small $\epsilon > 0$.

When $\boldsymbol{x}$ and $\boldsymbol{x}'$ are two randomly chosen data points there is a natural symmetry between insertion and deletion. In many settings however, one of the points $\boldsymbol{x}$ or $\boldsymbol{x}'$ is not an actual observation but is instead a reference value such as the gray images discussed

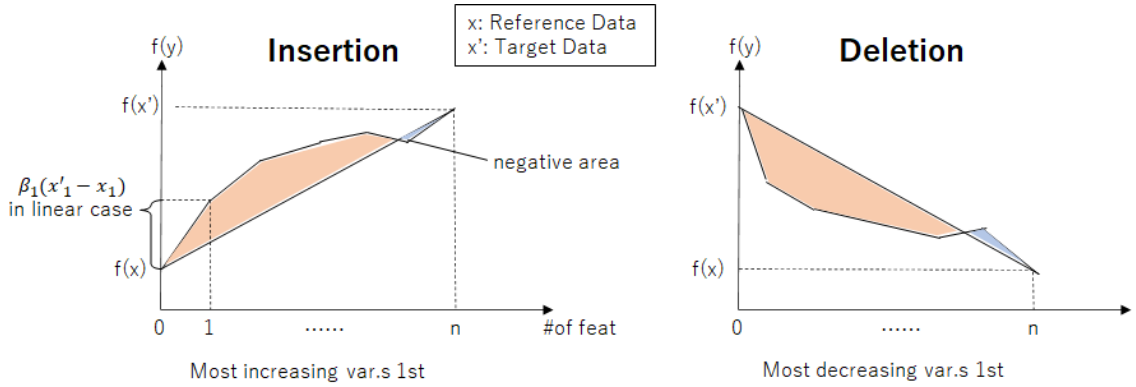

Figure 2: The left panel shows the (signed) area between a straight line and the curve formed by changing components of $\boldsymbol{x}'$ to those of $\boldsymbol{x}$ after ordering by their estimated positive impact on f. The right panel shows a signed area for deletion of variables.

above. As mentioned above, choices of $\boldsymbol{x}$ and $\boldsymbol{x}'$ to pair with each other are called policies. Section 5.2 has some example policies on our illustrative data. We include a counterfactual policy with motivation similar to counterfactual XAI. The relation between counterfactual XAI and choice of background data are also discussed in Albini et al. (2022) in detail.

A formal description of the curve is as follows. For a permutation $(\pi(1), \pi(2), \ldots, \pi(n))$ of $(1, 2, \ldots, n)$, and $1 \leqslant j \leqslant n$ define $\Pi(j) = \{\pi(1), \ldots, \pi(j)\}$ with $\Pi(0) = \varnothing$. Now let $\tilde{\boldsymbol{x}}^{(j)} = \boldsymbol{x}'_{\Pi(j)}{:}\boldsymbol{x}_{-\Pi(j)}$, that is

$$\tilde{\boldsymbol{x}}^{(j)}_k = \begin{cases} \boldsymbol{x}'_k, & k \in \Pi(j) \\ \boldsymbol{x}_k, & \text{else.} \end{cases}$$

For our theoretical study it is convenient to define

$$\text{AUC} = \sum_{j=0}^{n} f(\tilde{\boldsymbol{x}}^{(j)}). \tag{1}$$

If we connect the points $(j, f(\tilde{\boldsymbol{x}}^{(j)})$ by line segments then the area we get is a sum of trapezoidal areas

$$\sum_{j=1}^{n} \frac{1}{2}\big(f(\tilde{\boldsymbol{x}}^{(j-1)}) + f(\tilde{\boldsymbol{x}}^{(j)})\big) = \text{AUC} - \frac{1}{2}\big(f(\tilde{\boldsymbol{x}}^{(0)}) + f(\tilde{\boldsymbol{x}}^{(n)})\big).$$

The difference between this trapezoidal area and (1) is unaffected by the ordering permutation π because $\tilde{\boldsymbol{x}}^{(0)} = \boldsymbol{x}$ and $\tilde{\boldsymbol{x}}^{(n)} = \boldsymbol{x}'$ are invariant to the permutation. One could similarly omit either $j = 0$ or $j = n$ (or both) from the sum in (1) without changing the difference between areas attributed to any two permutations.

Our primary measure is the area below the curve but above a straight line from $(0, f(\tilde{\boldsymbol{x}}^{(0)}))$ to $(n, f(\tilde{\boldsymbol{x}}^{(n)}))$. It is the (signed) area between those curves, that is

$$\begin{aligned}\mathrm{ABC} &= \mathrm{AUC} - \mathrm{AUL}, \quad \text{where} \\ \mathrm{AUL} &= \frac{n+1}{2}\big(f(\tilde{\boldsymbol{x}}^{(0)}) + f(\tilde{\boldsymbol{x}}^{(n)})\big) = \frac{n+1}{2}\big(f(\boldsymbol{x}) + f(\boldsymbol{x}')\big)\end{aligned} \tag{2}$$

is a measure of the area under the straight line connecting $(0, f(\boldsymbol{x}))$ to $(n, f(\boldsymbol{x}'))$ compatible with our AUC formula from (1). The difference between ABC and AUC is also unaffected by the ordering of variables. The AUC and ABC going from $\boldsymbol{x}$ to $\boldsymbol{x}'$ is the same as that from $\boldsymbol{x}'$ to $\boldsymbol{x}$. That is, it only depends on the two selected points. The reason for this is that $\tilde{\boldsymbol{x}}^{(k)}$ going from $\boldsymbol{x}'$ to $\boldsymbol{x}$ equals $\tilde{\boldsymbol{x}}^{(n-k)}$ when we go from $\boldsymbol{x}$ to $\boldsymbol{x}'$. For the same reason the deletion areas are the same in both directions, but generally not equal to their insertion counterparts.

Both AUC and ABC have the same units that f has. Then for instance if f is measured in dollars then ABC/n is in dollars explained per feature. This normalization is different from that of Petsiuk et al. (2018) whose curve is over the interval $[0, 1]$ instead of $[0, n]$ and whose AUC is then interpreted in terms of a proportion of pixels.

3.2 Additive Functions

The AUC measurements above are straightforward to interpret when $f(\boldsymbol{x})$ takes the additive form $f_\varnothing + \sum_{j=1}^n f_j(x_j)$ for a constant $f_\varnothing$ and functions $f_j : \mathcal{X}_j \to \mathbb{R}$. We then easily find that

$$\mathrm{AUC} = (n+1)f(\boldsymbol{x}) + \sum_{j=1}^{n}(n-j+1)\big(f_j(x_j') - f_j(x_j)\big), \quad \text{and} \tag{3}$$

$$\mathrm{ABC} = \sum_{j=1}^{n}\Big(\frac{n+1}{2} - j\Big)\big(f_j(x_j') - f_j(x_j)\big). \tag{4}$$

The best ordering is, unsurprisingly, the one that sorts j in decreasing values of $f_j(x_j) - f_j(x_j')$. If f is additive then the insertion and deletion ABCs are the same. Also, the Shapley value for variable j is proportional to $f_j(x_j') - f_j(x_j)$ and so ordering variables by decreasing Shapley value maximizes the AUC and ABC.

3.3 Interactions

The effect of interactions is more complicated, but we only need interactions involving points $\boldsymbol{x}$ and $\boldsymbol{x}'$. We define the differences and iterated differences of differences that we need via

$$\begin{aligned}\Delta_j &= \Delta_j(\boldsymbol{x}, \boldsymbol{x}', f) = f(\boldsymbol{x}_j'{:}\boldsymbol{x}_{-j}) - f(\boldsymbol{x}), \quad \text{and} \\ \Delta_u &= \Delta_u(\boldsymbol{x}, \boldsymbol{x}', f) = \sum_{v\subseteq u}(-1)^{|u-v|} f(\boldsymbol{x}_v'{:}\boldsymbol{x}_{-v})\end{aligned}$$

for $u \subseteq 1{:}d$ with $\Delta_\varnothing = f(\boldsymbol{x})$ corresponding to no differencing at all. From Theorem 3 of Appendix B we get

$$\mathrm{AUC} = \sum_{u\subseteq 1:n}\big(n - \lceil u\rceil + 1\big)\Delta_u.$$

We can interpret this as follows: the interaction for variables u when represented as differences of differences, takes effect in its entirety once the last element j of u has been changed from x_j to x'_j. It then contributes to $n - \lceil u \rceil + 1$ of the summands. Thus, in addition to ordering the main effects from largest to smallest, the quality score for a permutation takes account of where large positive and large negative interactions are placed.

It is easy to see from (4) that for an additive function and a uniformly random permutation π we have $\mathbb{E}(\mathrm{ABC}) = 0$ because under such random sampling the expected rank of variable j is $(n+1)/2$.

Now suppose that we permute the variables 1 through n into a random permutation π. A fixed subset $u = (j_1, \ldots, j_{|u|})$ is then mapped to $\pi(u) = (\pi(j_1), \ldots, \pi(j_{|u|}))$. Under this randomization

$$\mathbb{E}(\mathrm{AUC}) = \sum_{u \subseteq 1:n} \big(n - \mathbb{E}(\lceil \pi(u) \rceil) + 1\big)\Delta_u f.$$

The next proposition gives $\mathbb{E}(\lceil u \rceil)$.

Proposition 1. *For $n \geqslant 1$ and $u \subseteq 1{:}n$ let $\pi(u)$ be a simple random sample of $|u|$ elements from $1{:}n$. Then*

$$\mathbb{E}(\lceil \pi(u) \rceil) = \frac{|u|(n+1)}{|u|+1}. \tag{5}$$

Proof The result holds trivially for $|u| = 0$, so we suppose that $|u| \geqslant 1$. For $k \in \{|u|, \ldots, n\}$, $\Pr(\lceil \pi(u) \rceil \leqslant k) = \binom{k}{|u|} / \binom{n}{|u|}$ because there are $\binom{n}{|u|}$ equally probable ways to select the elements of $\pi(u)$ and $\binom{k}{|u|}$ of those have $\lceil \pi(u) \rceil \leqslant k$. Subtracting $\Pr(\lceil \pi(u) \rceil \leqslant k-1)$ we get

$$\Pr(\lceil \pi(u) \rceil = k) = \binom{k-1}{|u|-1} \Big/ \binom{n}{|u|} = \frac{|u|}{k}\binom{k}{|u|} \Big/ \binom{n}{|u|}.$$

Then using the hockey stick identity,

$$\mathbb{E}(\lceil \pi(u) \rceil) = |u| \binom{n}{|u|}^{-1} \sum_{k=|u|}^{n} \binom{k}{|u|} = u \binom{n+1}{|u|+1} \Big/ \binom{n}{|u|} = \frac{|u|(n+1)}{|u|+1}.$$

■

Next we work out the expected value of AUC. Using the decomposition in Appendix B.1 we have

$$\mathrm{AUL} = \frac{n+1}{2}\big(f(\boldsymbol{x}) + f(\boldsymbol{x}')\big) = (n+1)\Delta_\varnothing + \frac{n+1}{2}\sum_{u \neq \varnothing} \Delta_u.$$

Then with Proposition 1 we find that

$$\begin{aligned}\mathbb{E}(\mathrm{ABC}) &= \mathbb{E}(\mathrm{AUC}-\mathrm{AUL})\\ &= \sum_{u\neq\varnothing}\Big(\frac{n+1}{2}-\mathbb{E}(\lceil\pi(u)\rceil)\Big)\Delta_u f\\ &= \sum_{u\neq\varnothing}\Big(\frac{n+1}{2}-\frac{|u|(n+1)}{|u|+1}\Big)\Delta_u f\\ &= \frac{n+1}{2}\sum_{u\neq\varnothing}\frac{1-|u|}{|u|+1}\Delta_u f.\end{aligned}$$

As noted above, $\mathbb{E}(\mathrm{ABC})=0$ if f has no interactions because $\mathbb{E}(\lceil\{j\}\rceil)=(n+1)/2$, but otherwise it need not be zero because $\mathbb{E}(\lceil u\rceil)>(n+1)/2$ for $|u|>1$. The contribution to ABC from a given interaction has the opposite sign of that interaction because $|u|-1<0$ for $|u|\geqslant 2$. We show in Appendix B.1 that $\mathbb{E}(\mathrm{ABC}+\mathrm{ABC}')=0$ under random permutation of the indices.

3.4 AUC Versus Shapley

If we order variables in decreasing order by Shapley value, that does not necessarily maximize the AUC. We can see this in a simple setup for $n=3$ by constructing certain values of Δ_u. We will exploit the delay $\lceil u\rceil$ with which an interaction gets 'credited' to an AUC to find our example.

Consider a setting with $n=3$ and $\Delta_1=3$, $\Delta_2=2$, $\Delta_3=1$, $\Delta_{\{1,2\}}=A$ to be chosen later and all other $\Delta_u=0$. Because the Δ_u values are also Harsanyi dividends (Harsanyi, 1959), the Shapley value shares them equally among their members. Therefore

$$\phi_1=3+A/2,\quad \phi_2=2+A/2\quad\text{and}\quad \phi_3=1.$$

So long as $A>-2$, the Shapley values are ordered $\phi_1>\phi_2>\phi_3$. The AUC for this ordering is

$$\mathrm{AUC}((1,2,3))=3\Delta_1+2\Delta_2+\Delta_1+(3-\lceil\{1,2\}\rceil+1)A=14+2A.$$

If we order the variables $(1,3,2)$ then the AUC is

$$\mathrm{AUC}((1,3,2))=3\Delta_1+2\Delta_3+\Delta_2+(3-\lceil\{1,3\}\rceil+1)A=13+A.$$

As a result we see that if $-2<A<-1$, then

$$\phi_2>\phi_3\quad\text{but}\quad \mathrm{AUC}((1,3,2))>\mathrm{AUC}((1,2,3)).$$

It is easy to show that there can be no counterexamples for $n=2$.

4. Incorporating Binary Features into Integrated Gradients

IG avoids the exponential computational costs that arise for Shapley value. However, as defined it is only available for variables with continuous values. Many problems have binary variables and so we describe here some approaches to including them.

IG is based on the Aumann-Shapley value from Aumann and Shapley (1974) who present an axiomatic derivation for it. We omit those axioms. See Sundararajan et al. (2017) and Lundberg and Lee (2017) for the axioms in a variable importance context.

We orient our discussion around Figure 3 that shows a setting with 3 variables. Panel (a) shows a target data point that differs from a baseline point in three coordinates. Panel (b) shows the diagonal path taken by integrated gradients. The gradient of $f(\boldsymbol{x})$ is integrated along that path to get the IG attributions:

$$A_f(\boldsymbol{x}, \boldsymbol{x}') \equiv \int_0^1 \nabla f(\boldsymbol{x} + t(\boldsymbol{x}' - \boldsymbol{x}))\,\mathrm{d}t \in \mathbb{R}^n.$$

If one of the variables is binary then one approach, shown in panel (c) is to simply jump from one value to another at some intermediate point, such as the midpoint. For differentiable f the integral of the gradient along the line segment given by the jump would, by the fundamental theorem of calculus, be the difference between f at the ends of that interval (times the Euclidean basis vector $(0, \ldots, 0, 1, 0, \ldots, 0)$ corresponding to that variable). Such a difference is computable for binary variables even though the points on the path are ill defined. Finally, the vector of Shapley values is an average over $n!$ paths making jumps from baseline to target in all possible variable orders (Sundararajan et al., 2017). Panel (d) shows two of those paths. For differentiable f one could integrate gradients along those paths as a way to compute the jumps, again by the fundamental theorem of calculus, and then average the path integrals.

Now suppose that we have $m > 0$ binary variables in a set $v \subset 1{:}n$. Without loss of generality suppose that $v = 1{:}m$. Then any data $\boldsymbol{x}$ and $\boldsymbol{x}'$ are in $\{0,1\}^m \times \mathbb{R}^{n-m}$ and for IG we need to consider arguments to f in $\mathbb{R}^n$. We consider three choices that we describe in more detail below:

a) Use the fitted f as if the binary $x_j \in [0,1]$, for $j \in 1{:}m$.

b) Replace f by a multilinear interpolation

$$g(\boldsymbol{x}) = \sum_{u \subseteq 1:m} f(\mathbf{1}_u{:}\mathbf{0}_{1:m-u}{:}\boldsymbol{x}_{(m+1):n}) \prod_{j \in u} x_j \prod_{j \in 1:m-u} (1 - x_j)$$

and compute the integrated gradients of g.

c) Take paths that jump for binary x_j as shown in Figure 3(c).

We call these choices, casting, interpolating and jumping, respectively.

Option **a** is commonly available as many machine learning models cast binary variables to real values when fitting. Option **b** interpolates: if $\boldsymbol{x}_{1:m} \in \{0,1\}^m$ then $g(\boldsymbol{x}) = f(\boldsymbol{x})$. Sundararajan et al. (2017) show that integrated gradients match Shapley values for functions that are a sum of a multilinear interpolation like g above plus a differentiable additive function. Unfortunately, the cost of evaluating $g(\boldsymbol{x})$ is $\Omega(2^m)$, which is exponential in the number of binary inputs. For option **c** we have to choose where to make the m jumps. For $m = 1$ we would naturally jump half way along the path though there is not an axiomatic reason for that choice. For $m > 1$ we have to choose m points on the curve at which to jump. Even if we decide that all of those jumps should be at the midpoint, we are left with $m!$ possible orders in which to make those jumps. By symmetry we might want to average over those orders but that produces a cost which is exponential in m.

Based on the above considerations we think that the best way to apply IG to binary variables is also the simplest. We cast the corresponding booleans to floats.

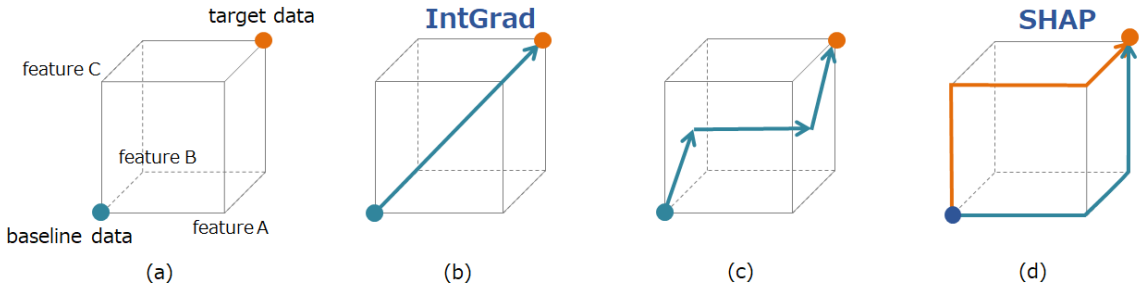

Figure 3: (a): $\boldsymbol{x}$ and $\boldsymbol{x}'$ placed in feature space represented as a cube, (b): usual Integrated Gradients with a straight line path connecting x and x', (c): Integrating along a path with one jump, (d): Shapley value expressed as an average of path integrals over $n!$ paths.

5. Experimental Results

In this section we illustrate insertion and deletion tests for regression. We compare our variance importance measures on two tabular datasets. The first one is about predicting the value of houses in India. It has mostly binary predictors and two continuous ones. The second dataset is from CERN and it computes the invariant mass produced from some electron collisions.

The methods we compare are Kernel SHAP (Lundberg and Lee, 2017), integrated gradients (Sundararajan et al., 2017), DeepLIFT (Shrikumar et al., 2017), Vanilla Grad (Simonyan et al., 2013), Input×Gradient (Shrikumar et al., 2016) and LIME (Ribeiro et al., 2016). Ancona et al. (2017) make a detailed study of backpropagation-based gradient methods including all of the above ones (this excludes LIME) as well as layer-wise relevance propagation (LRP) of Bach et al. (2015) that we did not include in our computations.

5.1 Bangalore Housing Data

The dataset we use lists the value in Indian rupees (INR) for houses in India. The data are from Kaggle at this URL:
`www.kaggle.com/ruchi798/housing-prices-in-metropolitan-areas-of-india`.
We use 38 of the 39 other columns in this dataset (excluding the "Location" column that contains various place names for simplicity), and we treat the "Area" and "No. of Bedrooms" columns as continuous variables.

We use only the data from Bangalore (6,207 records). Most of those data points were missing almost every predictor. We use only 1,591 complete data points. We normalized the output value by dividing by 10,000,000 INR. We centered the continuous variables at their means and then divided them by their standard deviations. We selected 80% of the data points at random to train a multilayer perceptron (MLP). The hyperparameters such as number of layers and ratio of dropouts are determined from a search described in Appendix A.2.

For the 20% of points that were held out (391 observations) we computed the ABC from (2) using our collection of variable importance methods. We also included a random variable ordering as a check. For each of those points $\boldsymbol{x}$ we made a careful selection of a reference point $\boldsymbol{x}'$ from holdout points as follows:

- the point $\boldsymbol{x}'$ had to differ from $\boldsymbol{x}$ in at least 12 features,
- it had to be among the smallest 20 such values of $\|\boldsymbol{x} - \cdot\|$, and
- among those 20 it had to have the greatest response difference $|f(\boldsymbol{x}) - f(\boldsymbol{x}')|$.

Having numerous different features makes the attribution problem more challenging. Having $\|\boldsymbol{x} - \boldsymbol{x}'\|$ small brings less exposure to the problems of unrealistic data hybrids. Finally, having large $|f(\boldsymbol{x}) - f(\boldsymbol{x}')|$, the absolute value of the sum of feature attributions for XAI algorithms with the completeness axiom, identifies data pairs in most need of an attribution. Despite having close feature vectors those pairs have quite different predicted responses.

Our implementation of KS used 120,000 samples. Our implementation of IG used a Riemann sum on 500 points to approximate the line integrals. The hyperparameters for other XAI methods are summarized in Appendix A.3.

The ABCs and their differences are summarized in Table 1. There we see numerically that KS was best for the insertion ABC and IG was second best. For the deletion measure it was essentially a three way tie for best among KS, IG and DeepLift.

The simple gradient based methods were disappointing. In particular, vanilla grad was worse than random. We note that vanilla grad uses a default variable scaling. We used the standard deviation of each input while another choice is to scale each variable to the interval $[0, 1]$. Neither of these choices use the specific baseline-target pair and this could cause poor performance.

The difference between KS and IG was not very large. Thus even in this setting where there are lots of binary predictors, IG was able to closely mimic KS. We see in Table 1 that insertion ABCs are on average higher than deletion ABCs for this policy.

While KS and IG and LIME and DeepLIFT make use of reference values in computation of feature attributions, vanilla grad and Input×Gradient do not require one to specify reference values. They are determined only by local information around the target data. Since our ABC criterion is defined in terms of reference values it is not surprising that methods which use those reference values get larger ABC values. It is interesting that a method like Input×Gradient that does not even know the baseline we compare to can do as well as it does here. We note that DeepLIFT does reasonably well compared to KS and IG, even though DeepLIFT is derived without any axiomatic properties such as the Aumann-Shapley axioms. Ancona et al. (2017) pointed out a connection wherein DeepLIFT can be interpreted as the approximation of a Riemann sum of IG by a single step with average value in spite of the difference in their computational procedures. Their implementation details are also summarized in Appendix A.3.

KS performed well and IG is a fast approximation to it. Therefore we compare the ABC of insertion and deletion tests for KS and IG in Figure 4. In both cases the left panel shows that the ABC for KS has a long tail. This is also true for IG, but to save space we omit that histogram. Instead we show in the middle panels that the ABC for IG is almost the same as that for KS point by point, with KS usually attaining a somewhat better (larger) ABC than IG. The right panels there show that ABCs for random orderings have nearly

Test Mode	Method	Mean	Std. Error
Insertion	Kernel SHAP	0.628	0.034
	Integrated Gradients	0.572	0.033
	DeepLIFT	0.548	0.032
	Vanilla Grad	−0.093	0.026
	Input×Gradient	0.206	0.027
	LIME	0.499	0.028
	Random	−0.020	0.023
	Kernel SHAP − Integrated Gradients	0.057	0.007
Deletion	Kernel SHAP	0.423	0.032
	Integrated Gradients	0.422	0.032
	DeepLIFT	0.425	0.031
	Vanilla Grad	−0.098	0.029
	Input×Gradient	0.185	0.029
	LIME	0.395	0.032
	Random	−0.023	0.012
	Kernel SHAP − Integrated Gradients	0.001	0.003

Table 1: Mean insertion and deletion ABCs for 391 of the Bangalore housing data points, rounded to three places.

symmetric distributions with insertion tests having a few more outliers than deletion tests do.

Figure 5 shows some insertion and deletion curves comparing a randomly chosen data point to a counterfactual reference point. Figure 6 shows analogous plots for the data with the greatest differences in ABC between KS and IG. It shows that a very large ABC difference between methods in the insertion test need not have a large difference in the deletion tests and vice versa.

5.2 CERN Electron Collision Data

The CERN Electron Collision Data (McCauley, 2014) is a dataset about dielectron collision events at CERN. It includes continuous variables representing the momenta and energy of the electrons, as well as discrete variables for the charges of the electrons (± 1: positrons or electrons). Only the data whose invariant mass of two electrons (or positrons) was in the range from 2 to 110 GeV were collected. We treat it as a regression problem to predict their invariant mass from the other 16 features.

The data contains the physical observables of two electrons after the collisions whose tracks are reconstructed from the information captured in detectors around the beam. The features are as follows: The total energy of the two electrons, the three directional momenta, the transverse momentum, the pseudorapidity, the phi angle and the charge of each electron. They are highly dependent features because some of them are calculated from the others. For instance, since a beam line is aligned on the z-axis as usual in particle physics, the

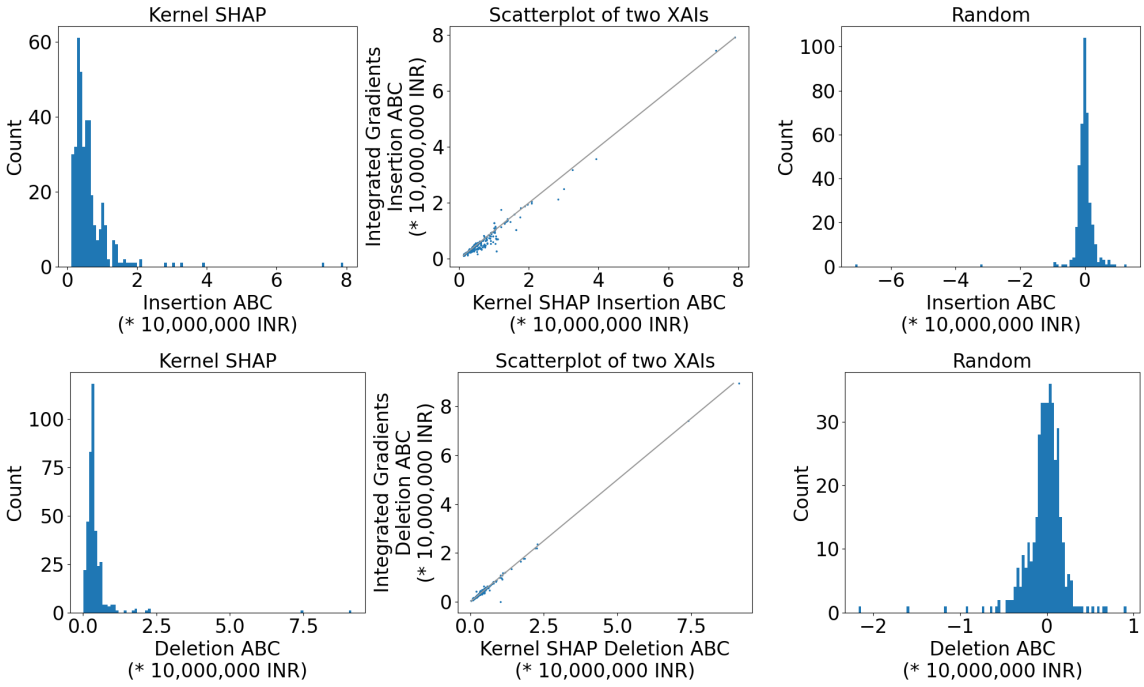

Figure 4: The top row shows results of insertion tests. The bottom row is for deletion tests.

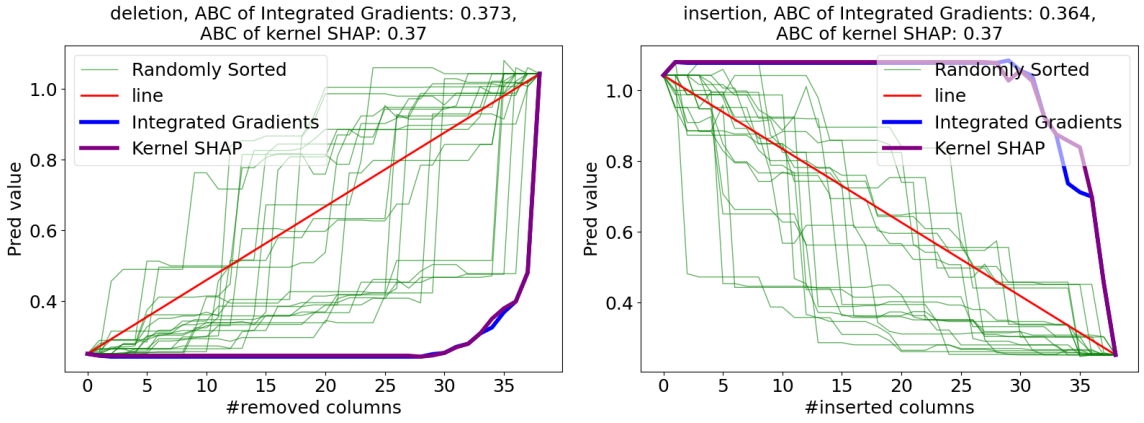

Figure 5: An example of the deletion and insertion tests for the Bangalore housing dataset. The left panel is for a deletion test and the right panel is for a insertion test. Both pictures include 20 curves for random variable orders.

transverse momenta p_{t_i} for $i = 1, 2$ are composed of p_{x_i} and p_{y_i} such that $p_{t_i}^2 = p_{x_i}^2 + p_{y_i}^2$. The phi angle ϕ_i is the angle between p_{x_i} and p_{y_i} where $p_{t_i} = p_{x_i} \cos \phi_i$. The total energy of them is also calculated relativistically. Since the momenta are recorded in GeV unit, which overwhelms the static mass of electrons ($\sim$ 511 keV in natural unit), the total energies E_i

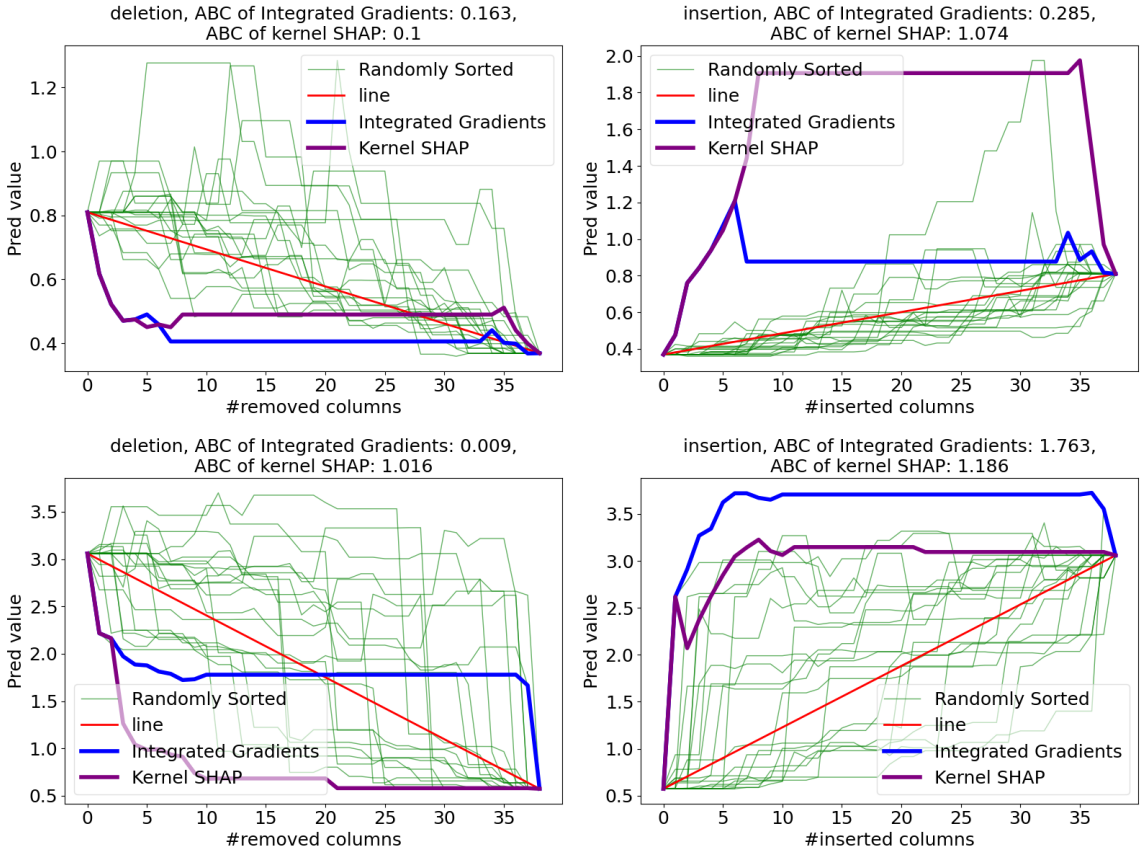

Figure 6: These figures show data points whose differences in ABC between KS and IG are largest in each of tests in the Bangalore housing dataset.

are $E_i^2 \simeq p_{x_i}^2 + p_{y_i}^2 + p_{z_i}^2$. The pseudorapidities η_i are given as angles from a beam line. The definition is $\eta = -\frac{1}{2} \ln \frac{|\boldsymbol{p}|-p_z}{|\boldsymbol{p}|+p_z} \simeq -\frac{1}{2} \ln \frac{E-p_z}{E+p_z}$ for each η_i. Regarding these definitions, only 8 (three directional momenta and the charges of each electrons) of the 16 are independent features, and the other features are used as convenient transformed coordinates in particle physics. Actually, the invariant mass M, the prediction target feature, is also approximated arithmetically from the momenta as $M^2 \simeq (E_1 + E_2)^2 - |\boldsymbol{p}_1 + \boldsymbol{p}_2|^2$ within 0.6% residual error on average for this dataset. From this viewpoint, users of machine learning might confirm that the models properly exclude the charges from evidence of the predictions via XAI. This aspect is, in a sense, a case where ground truth in XAI can be obtained from domain knowledge. We have made such investigations but omit them to save space and because they are not directly related to insertion and deletion testing.

We omit data with missing values, and randomly select 80% of the complete observations (79,931 data points) to construct an MLP. Embedding layers are not placed in this MLP, and all variables are Z-score normalized. Hyperparameters such as the number of units in each layer of the MLP are determined by a hyperparameter search described in Appendix A.4.

Test Mode	Methods	Mean	Std. Error
Insertion	Kernel SHAP	18.535	0.215
	Integrated Gradients	18.289	0.213
	DeepLIFT	18.118	0.211
	Vanilla Grad	−1.310	0.252
	Input×Gradient	7.620	0.216
	LIME	17.319	0.209
	Random	−0.380	0.175
	Kernel SHAP − Integrated Gradients	0.246	0.025
Deletion	Kernel SHAP	16.752	0.176
	Integrated Gradients	16.315	0.173
	DeepLIFT	16.646	0.176
	Vanilla Grad	0.226	0.256
	Input×Gradient	7.940	0.187
	LIME	15.845	0.170
	Random	−0.268	0.179
	Kernel SHAP − Integrated Gradients	0.437	0.025

Table 2: Mean insertion and deletion ABCs for 2000 CERN electron collision data points, under the counterfactual policy described in the text.

The predictions for the 2,000 held out data points $\boldsymbol{x}$ were inspected with both KS and IG. The reference data $\boldsymbol{x}'$ used in XAI methods are collected from these 2,000 data under this policy:

- $x'_j \neq x_j$ for all $j \in 1{:}n$ including charges,
- $\boldsymbol{x}'$ is among the 20 smallest such $\|\boldsymbol{x} - \cdot\|$ values, and
- it maximizes $|f(\boldsymbol{x}') - f(\boldsymbol{x})|$ subject to the above.

This policy is called the counterfactual policy below. It has similar motivations to the policy we used for the Bangalore housing data. In this case it was possible to compute KS exactly using $2^{16} = 65{,}536$ function evaluations.

The results are given in Table 2. KS is best for both insertion and deletion measures. IG and DeepLIFT are close behind. LIME is nearly as good and the simple gradient methods once again do poorly. As we did for the Bangalore housing data, we make graphical comparisons between KS and IG.

The results for the insertion test are shown in Figure 7. The deletion test results were very similar and are omitted. These results are similar to what we saw in the previous experiment. As a meaningful XAI metric, KS provides a larger ABC than the other orderings we tried. Also, even in this case where differentiability with respect to charges cannot be assumed, IG does nearly as well as KS.

Although most of data are close to the 45 degree line in the center panel of Figure 7 there are a few cases where KS gets a much larger ABC than IG does. Two such points are shown in Figure 8. Similarly to what we saw in the Bangalore housing example, the comparisons where KS and IG differ greatly in the insertion test has them similar in the deletion test

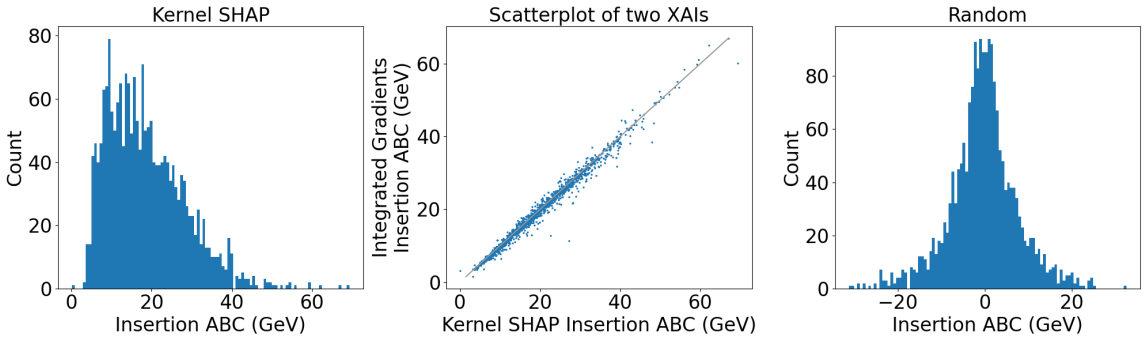

Figure 7: Results of the insertion test in CERN electron collision data. Results for the deletion test looked almost the same.

and vice versa. A scatterplot of deletion versus insertion areas is given in Figure 9. Here and in the following we again only pick KS and IG as representative examples. Most of the data are near the diagonal in that plot but there are some exceptional outlying points where the two ABCs are quite different from each other.

Next we consider two more policies, different from our counterfactual policy. In a 'one-to-one policy', observations are paired up completely at random. They almost always get different values of the continuous parameters and they get, on average one different value among the two binary values. We also consider an 'average policy' where the reference point has the average value for all features. Such points are necessarily unphysical, for instance, they have near zero charge.

The results of these two policies are shown in Table 3. KS attains the best ABC value in all four comparisons there and IG is always close but not always second, even though it treats the particle charges as continuous quantities. One striking feature of that table is that the insertion ABCs are much larger than the deletion ABCs. There is a simple explanation. The invariant masses must be positive and their distribution is positively skewed. The model never predicted a negative value and the predictions also have positive skewness. Because the predicted values satisfy a sharp lower bound, that reduces the maximum possible deletion ABC. Because they are positively skewed, higher insertion ABCs are possible. We see LIME does very well on the one-to-one policy examples as it did on the counterfactual policy, but it does not do well on the (unphysical) average policy comparisons.

Figure 10 shows results for the one-to-one policy. Unlike Figure 9 for the counterfactual policy the points for $f(\boldsymbol{x}) > f(\boldsymbol{x}')$ are exactly the same as those that had $f(\boldsymbol{x}') > f(\boldsymbol{x})$.

One data pair attaining an extreme difference in Figure 10 is inspected in Figure 11. This data gains over 150 GeV in the insertion ABC, which is anomalously large as this dataset is composed of events with invariant mass between 2 and 110 GeV. The figure shows that this large output is due to an extraordinary response to artificial data which appear on the path connecting $\boldsymbol{x}'$ and $\boldsymbol{x}$. Both XAI methods (IG and KS) correctly identify the features that bring on this effect for the insertion test. Since the one-to-one policy

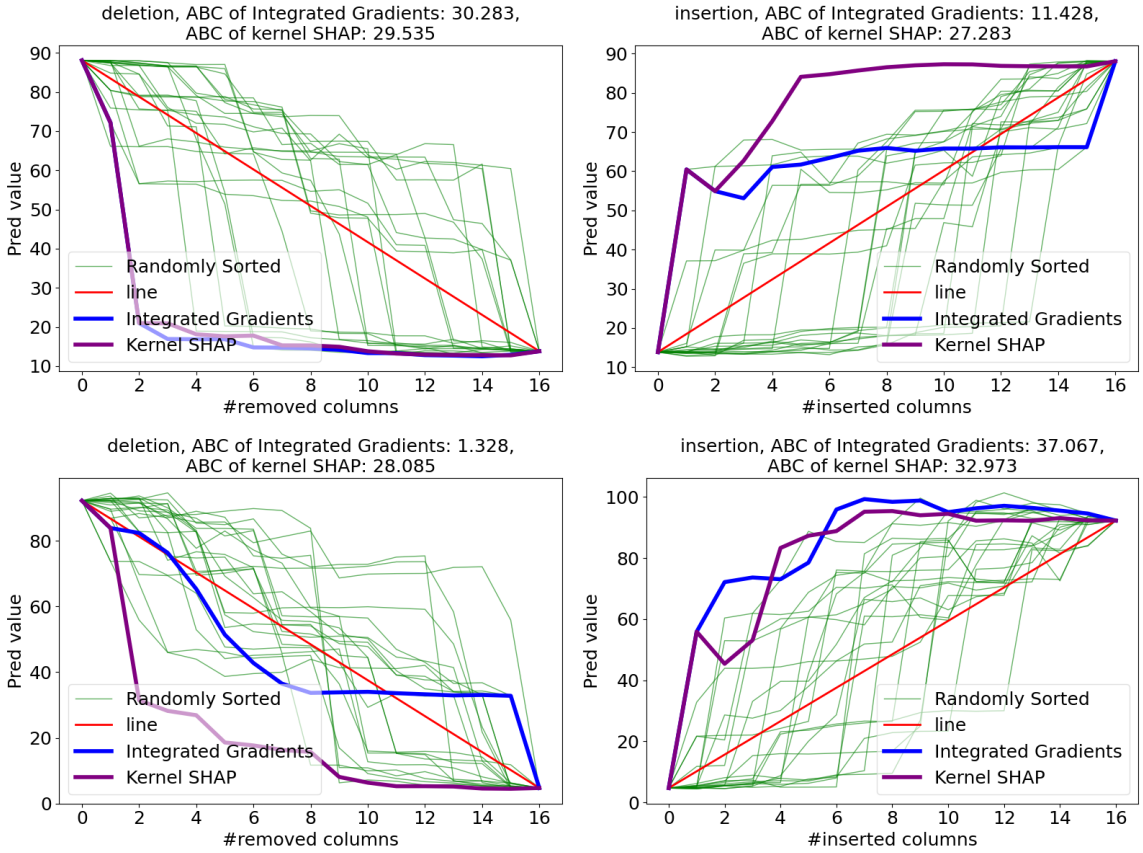

Figure 8: Deletion test outliers in the CERN electron collision data. The top row shows a comparison where KS had much greater ABC for insertion than IG had. The bottom row shows a comparison where KS had much greater ABC for deletion. In both cases KS was comparable to IG for the other ABC type.

can make long distance pairs compared to the counterfactual policy, synthetic data in the insertion and deletion processes can be far from the data manifold.

The result for the average policy where the reference data is common to all test data is shown in Figure 12. This is a setting where the reference data are very far out of distribution, just like a single color image is for an image classification task. In this case the deletion test, replacing real pixels in order by average ones, attains much smaller ABC values than the insertion test that starts with the average values. From the above results in Figure 9, 10 and 12 it is easy to observe that the relational distributions of insertion and deletion tests are totally different depending on the reference data policy, even for the same XAI algorithm. One cause is asymmetry in the response: if $f(\boldsymbol{x})$ is sharply bounded below but not above then insertion ABCs can be much larger than deletion ones.

The ABC including other XAI methods are aggregated in Table 3. The relationships among their magnitudes are almost same as those in the counterfactual policy of Table 2. It is also confirmed that insertion tests have larger ABC values than deletion tests with same

Test Mode	Method	One-to-One Mean	One-to-One Std. Error	Average Mean	Average Std. Error
Insertion	Kernel SHAP	30.392	0.535	23.518	0.216
	Integrated Gradients	28.430	0.515	21.549	0.251
	DeepLIFT	26.740	0.433	21.333	0.238
	Vanilla Grad	1.707	0.215	4.803	0.194
	Input×Gradient	14.677	0.397	21.610	0.236
	LIME	27.008	0.510	11.892	0.221
	Random	3.795	0.266	4.242	0.156
Deletion	Kernel SHAP	11.806	0.299	7.621	0.149
	Integrated Gradients	10.643	0.310	6.856	0.128
	DeepLIFT	11.485	0.294	7.305	0.134
	Vanilla Grad	−5.571	0.323	−4.025	0.135
	Input×Gradient	1.839	0.296	5.957	0.139
	LIME	10.877	0.269	1.420	0.160
	Random	−3.797	0.301	−4.323	0.157

Table 3: Mean insertion and deletion ABCs for 2,000 of the CERN Electron Collision Data points whose reference data are determined in one-to-one policy and average policy respectively. The figures are rounded to three places.

Policy	Number of Different Cols (Total Number = 16)	Correlation Coefficients KS	Correlation Coefficients IG
Counterfactual	16	0.820	0.799
One-to-One	14.991	−0.276	−0.297
Average	16	0.587	0.499

Table 4: For three policies on $(\boldsymbol{x}, \boldsymbol{x}')$ in the CERN data: the average number of j with $\boldsymbol{x}_j \neq \boldsymbol{x}'_j$ and the correlation between insertion and deletion ABCs, for both KS and IG.

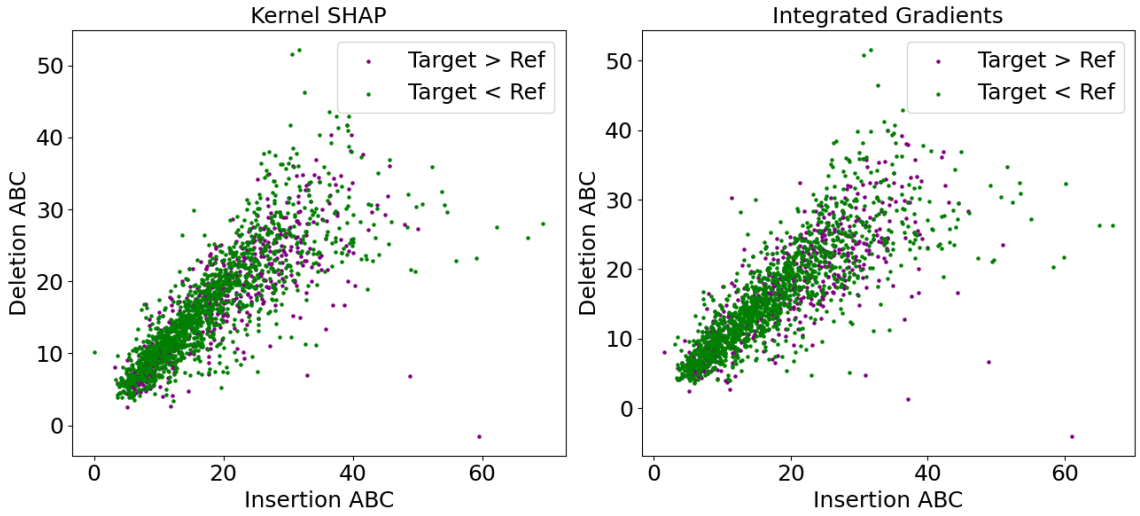

Figure 9: For the CERN data with the counterfactual policy, these figures show deletion versus insertion ABCs for KS (left) and IG (right). They are colored depending on relations of model outputs at target and reference point; Purple dots: $f(\boldsymbol{x}') > f(\boldsymbol{x})$ and green dots: $f(\boldsymbol{x}') < f(\boldsymbol{x})$.

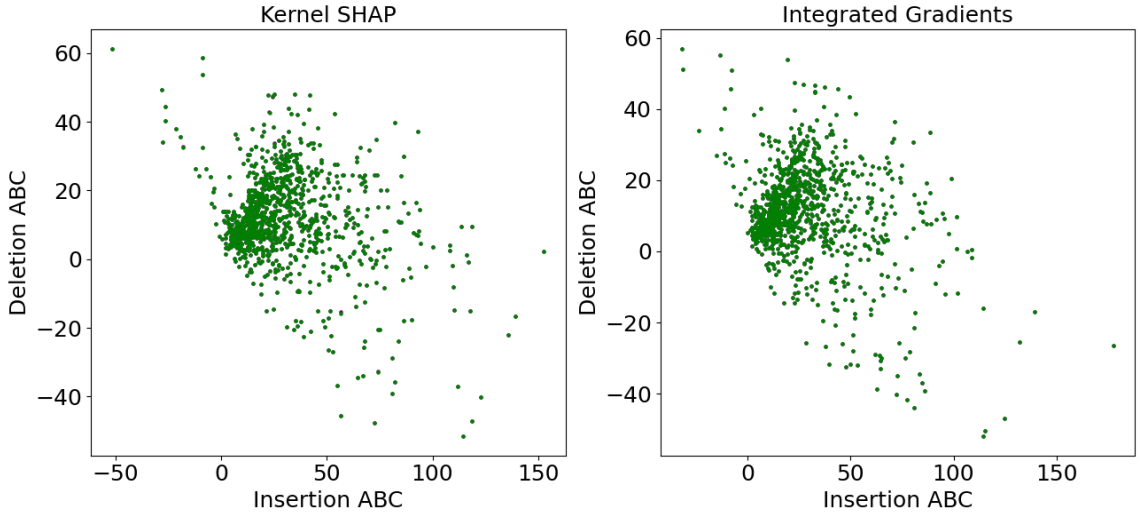

Figure 10: For the CERN data with the one-to-one policy, these figures plot deletion versus insertion ABCs. The left plot is for KS and the right is for IG. Each dot correspond to two data that configure a pair.

setup in general and their differences are larger than those of the counterfactual policy. This point supports our previous discussion about their asymmetry. The computation of

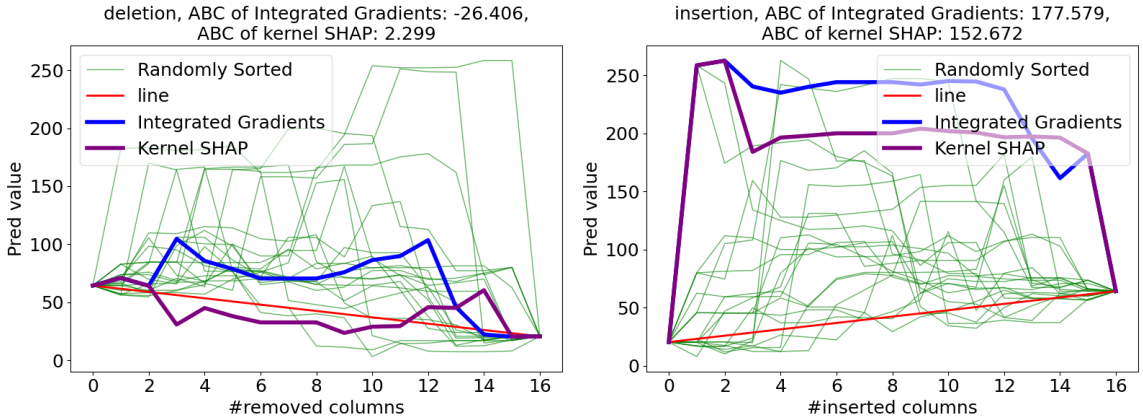

Figure 11: Insertion and deletion curves for the most asymmetrical data in Figure 10 that has over 150 GeV in insertion ABC and near to zero GeV in deletion ABC.

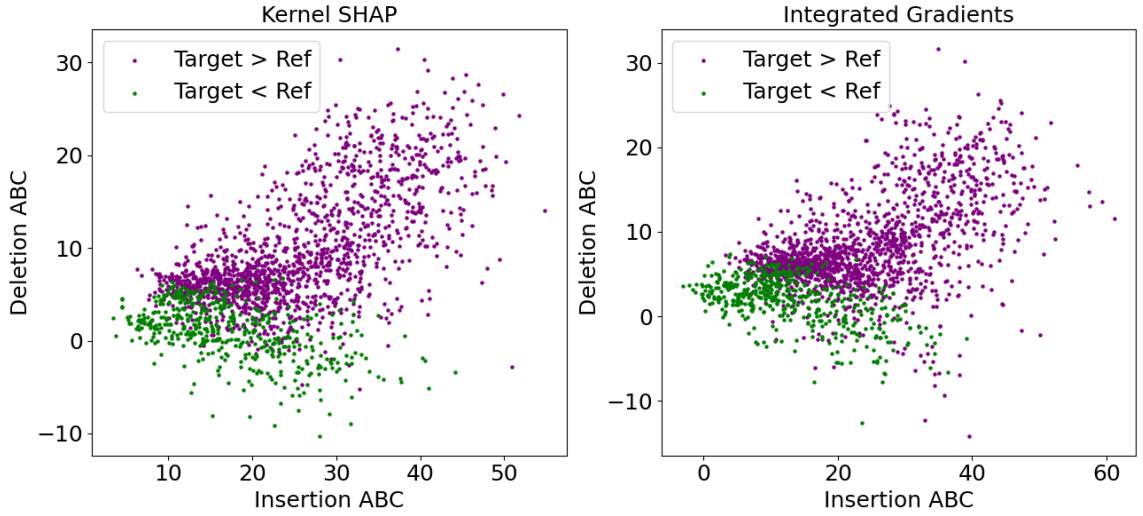

Figure 12: For the CERN data with the average policy, these figures plot deletion versus insertion ABCs. The left plot is for KS and the right is for IG.

Input×Gradient does not take the reference data into account, and this might explain why Input×Gradient has comparatively scores compared to other methods in the average policy.

The average number of different columns between reference data and target data and correlation coefficients of ABCs in two kinds of tests are summarized in Table 4. All sixteen columns have different values in the counterfactual policy by definition. The deviation from sixteen in the value in one-to-one policy is mostly from the two charges of the data, which can take only two levels, $+1$ or -1. In this sense, the reference data in average policy is unphysical data since it has charges that are near zero. The correlation coefficients between

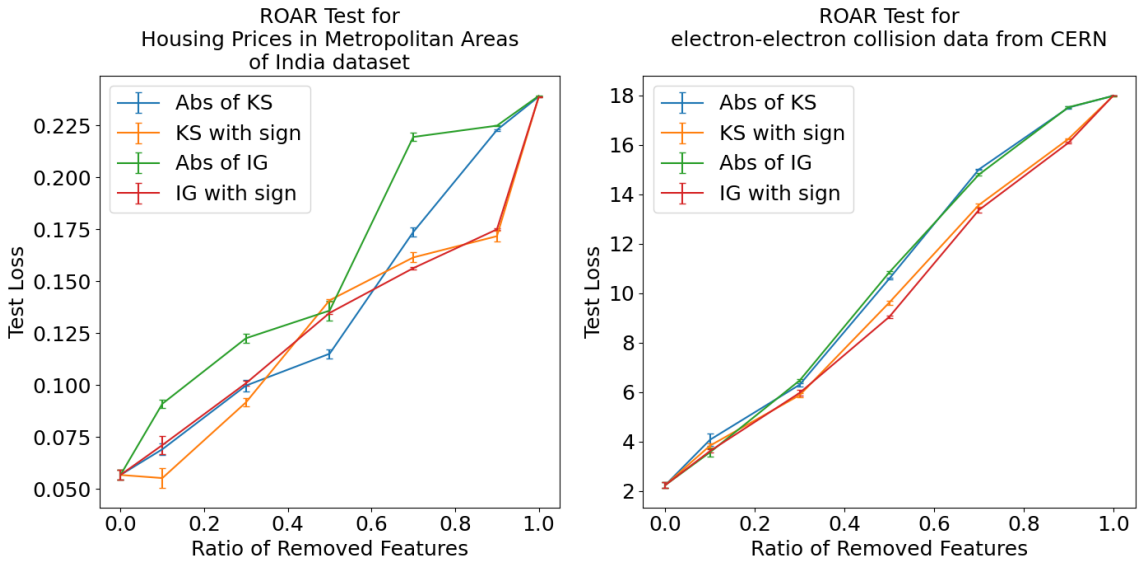

Figure 13: The result of ROAR test for the Bangalore housing dataset and the CERN Electron Collision Data.

two tests also vary between policies. We note also that the behavior of the two ABCs in the average policy strongly depends on whether $f(\boldsymbol{x}) > f(\boldsymbol{x}')$ or $f(\boldsymbol{x}) < f(\boldsymbol{x}')$ as seen in the scatter plots Figure 12.

6. RemOve and Retrain Methods

In this section we compare KS and IG via ROAR (RemOve And Retrain) (Hooker et al., 2019). ROAR is significantly more expensive to study than the other methods we consider as it requires retraining the models, and so we did not apply it to all of the methods. We opted to apply it just to Kernel SHAP and IG. We chose IG as our representative fast method because IG can be used on more general models than DeepLIFT can. We chose Kernel SHAP as the other method because it had best or near best ABC values on our numerical examples. The task in ROAR is about which variables are important to the model's accuracy and not about which variables are important to any specific prediction. As a result the values in ROAR are not comparable to the other AUC and ABC values that we have computed.

The original proposal of ROAR measures the drop in accuracy for image classification tasks and applying it to regression tasks with tabular data raises the same issues as extending insertion/deletion tests to regression. We measure the test loss on held out data as a measure of retrained model performance. Retraining procedures are taken with an increasing number of removed features at each quantile in $\{0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$, where 1.0 means that all features are removed. The model architecture and hyperparameters of retrained models are the original models as described in Sections A.2 and A.4.

To use ROAR we must decide how to remove features in the data. Removing in the original ROAR algorithm means padding pixels of the original images with noninformative values such as gray levels. In our experiments, the important features are overwritten by average values for continuous features and by modes for discrete features. These average and modal values are taken from the training data.

The results of our ROAR calculations are shown in Figure 13. Features to be removed are sorted both by their absolute values and by their original signed values for both KS and IG. The error bars show plus or minus one standard error computed from five replicates.

Since ROAR in this experiment measures the Huber loss on the test points, it should be unaffected by the signs of the attributions and sensitive to their magnitude. For this reason, sorting features by the absolute values of their attributions should give a better score than sorting by their signed values. This is a contrasting point to insertion and deletion tests.

The results in Figure 13 are surprising to us. The curves we see are very nearly straight lines connecting the loss with all features present to the loss with no features present, so there is very little area between them and a straight line connecting the end points. This means that the loss in accuracy from variables deemed most important is about the same as those deemed less important. This could be because neither KS nor IG are able to identify important predictors for this task. It could also be that the majority of predictor variables in this data can be replaced by a combination of some other predictors which then prevents a large reduction in accuracy from removing a subset of predictors prior to retraining.

A second surprise is that the signed ordering, that we used as a control that should have been beaten by the absolute ordering was nearly as good as the absolute ordering.

As before KS and IG are comparable, though here both seem disappointing. Using IG with variables sorted by their absolute values even came out superior to KS in the Bangalore housing dataset.

7. Conclusion

In this paper we have extended insertion and deletion tests to regression problems. That includes getting formulas for the effects of interactions on the AUC and ABC measures, finding the expected area between the insertion/deletion curve under random variable ordering, and replacing the horizontal axis by a more appropriate straight line reference. We gave a condition under which sorting variables by their Shapley value will optimize ABC as well as constructing an example where that does not happen.

We compared six methods and several policies on two datasets. We find that overall the Kernel SHAP gave the best areas. The much faster Integrated Gradients method was nearly as good. In order to even run IG in settings with binary variables, some strategy for using continuum values must be employed. We opted for the simplest choice of just casting the booleans to real values.

A very natural policy question is whether to prefer insertion or deletion. Petsiuk et al. (2018) consider both and do not show a strong preference for one over the other. They use deletion when comparing a real image to a blank image (deleting real pixels by replacing them with zeros). They use insertion when comparing a real image to a blurred one (inserting real pixels into the blurred image). In other words the choice between insertion and deletion is driven by the counterfactual point. In the regression setting both inputs points

could be real data. By studying both insertion and deletion we have seen that they can differ. A natural way to break the tie is to sum the ABC values for both insertion and deletion. Under a completely random permutation the expected value of that sum is zero. See Appendix B.1. In our examples, IG closely matches KS for both insertion and deletion, so it also matches their sum.

Acknowledgments

This work was supported by the U.S. National Science Foundation grants IIS-1837931 and DMS-2152780, and by Hitachi, Ltd. We thank Benjamin Seiler for helpful comments. Comments from three anonymous reviewers have helped us improve this paper.

References

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. Debugging tests for model explanations. *Advances in Neural Information Processing Systems*, 33:700–712, 2020.

Chirag Agarwal, Satyapriya Krishna, Eshika Saxena, Martin Pawelczyk, Nari Johnson, Isha Puri, Marinka Zitnik, and Himabindu Lakkaraju. OpenXAI: Towards a transparent evaluation of model explanations. *Advances in Neural Information Processing Systems*, 35:15784–15799, 2022.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

Emanuele Albini, Jason Long, Danial Dervovic, and Daniele Magazzeni. Counterfactual Shapley additive explanations. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1054–1070, 2022.

Ö. F. Aliş and H. Rabitz. Efficient implementation of high dimensional model representations. *Journal of Mathematical Chemistry*, 29(2):127–142, 2001.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. Technical report, arXiv:1711.06104, 2017.

Robert J. Aumann and Lloyd S. Shapley. *Values of Non-Atomic Games*. Princeton University Press, Princeton, NJ, 1974.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models. *Data Mining and Knowledge Discovery*, pages 1–60, 2023.

Yi Cai, Arthur Zimek, and Eirini Ntoutsi. XPROAX-local explanations for text classification with progressive neighborhood approximation. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2021.

A. P. Dawid and M. Musio. Effects of causes and causes of effects. *Annual Review of Statistics and Its Application*, 9, 2021.

B. Efron and C. Stein. The jackknife estimate of variance. *Annals of Statistics*, 9:586–596, 1981.

Thomas Fel, Remi Cadene, Mathieu Chalvidal, Matthieu Cord, David Vigouroux, and Thomas Serre. Look at the variance! efficient black-box explanations with Sobol-based sensitivity analysis. *Advances in Neural Information Processing Systems*, 34, 2021.

R. A. Fisher and W. A. Mackenzie. The manurial response of different potato varieties. *Journal of Agricultural Science*, xiii:311–320, 1923.

Tristan Gomez, Thomas Fréour, and Harold Mouchère. Metrics for saliency map evaluation of deep learning explanation methods. In *International Conference on Pattern Recognition and Artificial Intelligence*, pages 84–95. Springer, 2022.

John C Harsanyi. A bargaining model for cooperative n-person games. In *Contributions to the Theory of Games IV*, volume 2, pages 325–355. Princeton University Press, Princeton, NJ, 1959.

Johannes Haug, Stefan Zürn, Peter El-Jiz, and Gjergji Kasneci. On baselines for local feature attributions. Technical report, arXiv:2101.00905, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

W. Hoeffding. A class of statistics with asymptotically normal distribution. *Annals of Mathematical Statistics*, 19(3):293–325, 1948.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *Advances in neural information processing systems*, 32, 2019.

Cheng-Yu Hsieh, Chih-Kuan Yeh, Xuanqing Liu, Pradeep Ravikumar, Seungyeon Kim, Sanjiv Kumar, and Cho-Jui Hsieh. Evaluations and methods for explanation through robustness analysis. In *International Conference on Learning Representation (ICLR)*, 2021.

Aya Abdelsalam Ismail, Mohamed Gunady, Hector Corrada Bravo, and Soheil Feizi. Benchmarking deep learning interpretability in time series predictions. *Advances in neural information processing systems*, 33:6441–6452, 2020.

Cosimo Izzo, Aldo Lipani, Ramin Okhrati, and Francesca Medda. A baseline for Shapley values in MLPs: From missingness to neutrality. Technical report, arXiv:2006.04896, 2020.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch. Technical report, arXiv:2009.07896, 2020.

I Elizabeth Kumar, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle Friedler. Problems with Shapley-value-based explanations as feature importance measures. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, 2020.

F. Kuo, I. Sloan, G. Wasilkowski, and H. Woźniakowski. On decompositions of multivariate functions. *Mathematics of computation*, 79(270):953–966, 2010.

Q Vera Liao and Kush R Varshney. Human-centered explainable AI (XAI): From algorithms to user experiences. Technical report, arXiv:2110.10790, 2021.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Masayoshi Mase, Art B Owen, and Benjamin Seiler. Explaining black box decisions by Shapley cohort refinement. Technical report, arXiv:1911.00467, 2019.

Masayoshi Mase, Art B Owen, and Benjamin B Seiler. Variable importance without impossible data. Technical report, arXiv:2205.15750, 2022.

Thomas McCauley. Events with two electrons from 2010. CERN Open Data Portal., 2014.

Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable AI. *ACM Computing Surveys*, 2022.

Ryan O'Donnell. *Analysis of Boolean functions*. Cambridge University Press, New York, 2014.

Prathyush S Parvatharaju, Ramesh Doddaiah, Thomas Hartvigsen, and Elke A Rundensteiner. Learning saliency maps to explain deep time series classifiers. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1406–1415, 2021.

Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized input sampling for explanation of black-box models. Technical report, arXiv:1806.07421, 2018.

Elmar Plischke, Giovanni Rabitti, and Emanuele Borgonovo. Computing Shapley effects for sensitivity analysis. *SIAM/ASA Journal on Uncertainty Quantification*, 9(4):1411–1437, 2021.

Marco Tulio Ribeiro, 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*, pages 1135–1144, 2016.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Waddah Saeed and Christian Omlin. Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems*, 263:110273, 2023.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. Technical report, arXiv:1605.01713, 2016.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. Technical report, arXiv:1312.6034, 2013.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. Technical report, arXiv:1706.03825, 2017.

I. M. Sobol'. *Multidimensional Quadrature Formulas and Haar Functions*. Nauka, Moscow, 1969. (In Russian).

I. M. Sobol'. Theorems and examples on high dimensional model representation. *Reliability Engineering & System Safety*, 79(2):187–193, 2003.

Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020.

Mukund Sundararajan and Amir Najmi. The many Shapley values for model explanation. In *International Conference on Machine Learning*, pages 9269–9278. PMLR, 2020.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. *California Institute of Technology*, 2011.

Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

Appendix A. Detailed Model Descriptions of the Experiments

This appendix provides some background details on the experiments conducted in this article.

A.1 The Example in Image Classification

Here we summarize how the example of insertion and deletion tests in image classification shown in Figure 1 was computed. The image is from Wah et al. (2011) and the model is pretrained for classification in ImageNet (Russakovsky et al., 2015) whose architecture is EfficientNet-B0 (Tan and Le, 2019). The preprocessing of the model includes cropping the center of the image to make it square as shown in the saliency map of Figure 1.

The saliency map is computed using SmoothGrad of Smilkov et al. (2017). It averages IG computations over 300 randomly generated baseline images of Gaussian noise. The model output are distributed to the latent features in the first convolutional layer (implemented in Captum (Kokhlikyan et al., 2020) as layer integrated gradient). That layer has 32 blocks each of which has a 112×112 grid of 3 pixels times 3 pixels. The effect of any pixel is summed over those 32 blocks and over all 3×3 patterns that contain it. In the insertion and deletion test, this saliency map is then the same size, 224×224, as the preprocessed original image. The reference image for both insertion and deletion tests was a black image.

The AUCs for several different choices of reference data are summarized in Table 5. Note that the definition of AUCs are different from the main material of this paper and small deletion AUC is better in this situation. The reference image has a very significant effect on the AUCs.

A.2 Model for the Bangalore Housing Data

The detail of the model used in Section 5.1 is summarized in Table 6. Those hyperparameters, including the number of intermediate layers were obtained from a hyperparameter search using Optuna (Akiba et al., 2019). The model is optimized with Huber loss. Each intermediate layer is accompanied with parametric ReLU (He et al., 2015) and dropout layers. The dropout ratio is common in all of them.

Reference Image	Deletion AUC	Insertion AUC
Blurred	0.981	0.740
Mean	0.663	0.187
White	0.366	0.175
Black	0.270	0.201

Table 5: AUCs for the albatross example of Figure 1 using various reference images. The parameter for blurring the image is the same one Petsiuk et al. (2018) used.

Hyperparameter	Value
Dropout Ratio	0.10031
Learning Rate	1.7389×10^{-2}
Number of Neurons	[333–465–86–234]
Huber Parameter	1.0

Table 6: Parameters of the MLP model for the Bangalore housing data.

A.3 Other XAI methods in Tables 1, 2 and 3

The implementation details of the other XAI methods than KS and IG which appear in Tables 1, 2 and 3 are summarized in this subsection.

We use the implementations on Captum (Kokhlikyan et al., 2020) for those methods, DeepLIFT (Shrikumar et al., 2017), Vanilla Grad (Simonyan et al., 2013), Input×Gradient (Shrikumar et al., 2016) and LIME (Ribeiro et al., 2016) with default set arguments. Inputs of Input×Gradient are those after applying Z-score normalizing in the electron-electron collision data from CERN. Zeros in binary vectors are replaced by a small negative value (-10^{-4}) in Input×Gradient to avoid degeneration in the Metropolitan Areas of India dataset. As we use parametric ReLU as the activation functions in our model, it is also treated as a usual nonlinear function in the DeepLIFT calculations. The reference values that can be set in LIME and DeepLIFT are chosen as same data to KS and IG, depending on their policy.

A.4 CERN Electron Collision Data

The hyperparameters for the model used in Section 5.2 are given in Table 7. They were obtained from a hyperparameter search using Optuna (Akiba et al., 2019). Each intermediate layer is a parametric ReLU with dropout. The dropout ratio is common to all of the layers. The performance for test data is depicted in Figure 14. The model is overall very accurate but the very highest values are systematically underestimated.

Appendix B. Proof of Theorem 3

Here we prove that $\mathrm{AUC} = \sum_{u \subseteq 1:n} (n - \lceil u \rceil + 1) \Delta_u f$. We use the anchored decomposition that we define next. We also connect that decomposition to some areas of the literature.

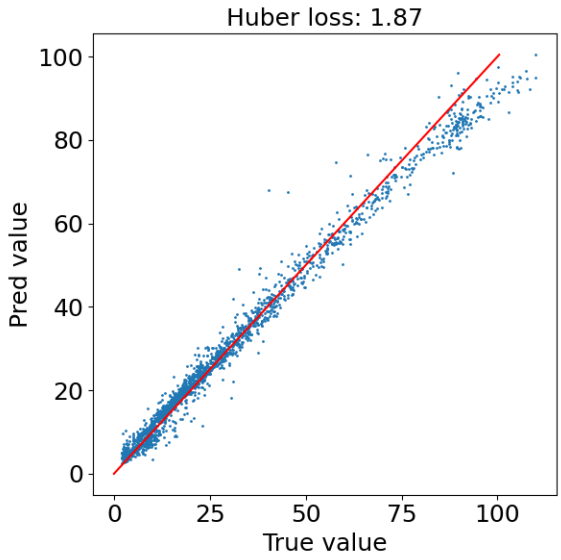

Figure 14: This figure plots estimated versus true invariant masses for held out points of the electron-electron collision data from CERN.

Hyperparameter	Value
Dropout Ratio	0.11604
Learning Rate	1.9163×10^{-4}
Number of Neurons	[509–421–65–368–122–477]
Huber Parameter	1.0

Table 7: Parameters of the MLP model for the CERN Electron Collision Data.

The anchored decomposition is a kind of high dimensional model representation (HDMR) that represents a function of n variables by a sum of 2^n functions, one per subset of $1{:}n$ where the function for $u \subseteq 1{:}n$ depends on $\boldsymbol{x}$ only through $\boldsymbol{x}_u$. The best known HDMR is the ANOVA of Fisher and Mackenzie (1923), Hoeffding (1948), Sobol' (1969) and Efron and Stein (1981) but there are others. See Kuo et al. (2010).

The anchored decomposition goes back at least to Sobol' (1969). It does not require a distribution on the inputs. Instead of centering higher order interaction terms by subtracting expectations, which don't exist without a distribution, it centers by subtracting values at default or anchoring input points. We only need it for functions on $\{0,1\}^n$ and without loss of generality we take the anchor to be all zeros.

We use $\mathbf{0}, \mathbf{1} \in \{0,1\}^n$ for vectors of all ones and all zeros, respectively. For $u \subseteq 1{:}n$ we write $e_u = \mathbf{0}_u{:}\mathbf{1}_{-u}$ generalizing the standard basis vectors e_j. The function we need to study is $g : \{0,1\}^n \to \mathbb{R}$ where

$$g(e_u) = f(\boldsymbol{x}_u{:}\boldsymbol{x}'_{-u}),$$

gives the values in the curve we study.

The anchored decomposition of $g : \{0,1\}^n \to \mathbb{R}$ is

$$\begin{aligned} g(\boldsymbol{z}) &= \sum_{u \subseteq 1:n} g_u(\boldsymbol{z}), \quad \text{with} \\ g_\varnothing(\boldsymbol{z}) &= g(\boldsymbol{0}), \quad \text{and for } |u| > 0, \\ g_u(\boldsymbol{z}) &= g(\boldsymbol{z}_u{:}\boldsymbol{0}_{-u}) - \sum_{v \subsetneq u} g_v(\boldsymbol{z}). \end{aligned}$$

The main effect in an anchored decomposition is $g_j(\boldsymbol{z}) = g(\boldsymbol{z}_j{:}\boldsymbol{0}_{-j}) - g(\boldsymbol{0})$ and the two factor term for indices $j \neq k$ is

$$\begin{aligned} g_{\{j,k\}}(\boldsymbol{z}) &= g(\boldsymbol{z}_{\{j,k\}}{:}\boldsymbol{0}_{-\{j,k\}}) - g_j(\boldsymbol{z}) - g_k(\boldsymbol{z}) - g_\varnothing(\boldsymbol{z}) \\ &= g(\boldsymbol{z}_{\{j,k\}}{:}\boldsymbol{0}_{-\{j,k\}}) - g(\boldsymbol{z}_j{:}c_{-j}) - g(\boldsymbol{z}_k{:}c_{-k}) + g(\boldsymbol{0}). \end{aligned}$$

There is an inclusion-exclusion-Möbius formula

$$g_u(\boldsymbol{z}) = \sum_{v \subseteq u} (-1)^{|u-v|} g_v(\boldsymbol{z}_v{:}\boldsymbol{0}_{-v}).$$

See for instance Kuo et al. (2010). The anchored decomposition is also called cut-HDMR (Aliş and Rabitz, 2001) in chemistry, and finite differences-HDMR in global sensitivity analysis (Sobol', 2003). When f is the value function in a Shapley value context, the values $g_u(\boldsymbol{1})$ are known as Harsanyi dividends (Harsanyi, 1959). Many of the quantities we use here also feature prominently in the study of Boolean functions $f : \{0,1\}^n \to \{0,1\}$ (O'Donnell, 2014).

The next Lemma is from Mase et al. (2019). We include the short proof for completeness.

Lemma 2. *For integer $n \geqslant 1$, let $f : \{0,1\}^n \to \mathbb{R}$ have the anchored decomposition $g(\boldsymbol{z}) = \sum_{u \subseteq 1:n} g_u(\boldsymbol{z})$. Then for $w \subseteq 1{:}n$,*

$$g_u(\boldsymbol{e}_w) = g_u(\boldsymbol{1}) 1_{u \subseteq w},$$

where $\boldsymbol{e}_w = \boldsymbol{1}_w{:}\boldsymbol{0}_{-w}$.

Proof The inclusion-exclusion formula for the binary anchored decomposition is

$$g_u(\boldsymbol{z}) = \sum_{v \subseteq u} (-1)^{|u-v|} g(\boldsymbol{z}_v{:}\boldsymbol{0}_{-v}).$$

Suppose that $z_j = 0$ for $j \in u$. Then, splitting up the alternating sum

$$g_u(\boldsymbol{z}) = \sum_{v \subseteq u-j} (-1)^{|u-v|} (g(\boldsymbol{z}_v{:}\boldsymbol{0}_{-v}) - f(\boldsymbol{z}_{v+j}{:}\boldsymbol{0}_{-v-j})) = 0$$

because $\boldsymbol{z}_v{:}\boldsymbol{0}_{-v}$ and $\boldsymbol{z}_{v+j}{:}\boldsymbol{0}_{-v-j}$ are the same point when $z_j = 0$. It follows that $g_u(\boldsymbol{e}_w) = 0$ if $u \not\subseteq w$.

Now suppose that $u \subseteq w$. First $g_u(\boldsymbol{z}) = g_u(\boldsymbol{z}_u{:}\mathbf{1}_{-u})$ because g_u only depends on $\boldsymbol{z}$ through $\boldsymbol{z}_u$. From $u \subseteq w$ we have $(\boldsymbol{e}_w)_u = \mathbf{1}_u$. Then $g_u(\boldsymbol{e}_w) = g_u(\mathbf{1}_u{:}\mathbf{1}_{-u}) = g_u(\mathbf{1})$, completing the proof. ■

We are now ready to state and prove our theorem expressing the AUC in terms of the anchored decomposition. Without loss of generality it takes π to be the identity permutation.

Theorem 3. *Let $f : \boldsymbol{x} \to \mathbb{R}$ and let $g : \{0,1\}^n \to \mathbb{R}$ be defined by $g(\boldsymbol{e}_u) = f(\boldsymbol{x}'_u{:}\boldsymbol{x}_{-u})$, and let $\pi = (1, 2, \ldots, n)$. Then the* AUC *given by* (1) *satisfies*

$$\mathrm{AUC} = \sum_{u \subseteq 1:n} g_u(\mathbf{1})(n - \lceil u \rceil + 1) = \sum_{u \subseteq 1:n} (n - \lceil u \rceil + 1)\Delta_u f. \tag{6}$$

Proof First, $\tilde{\boldsymbol{x}}^{(j)} = \boldsymbol{x}_{1:j}{:}\boldsymbol{x}'_{-\{1:j\}}$. Then

$$\mathrm{AUC} = \sum_{j=0}^{n} f(\tilde{\boldsymbol{x}}^{(j)}) = \sum_{j=0}^{n} g(e_{1:j}) = \sum_{j=0}^{n} \sum_{u \subseteq 1:n} g_u(e_{1:j}).$$

Next using Lemma 2, we find that AUC equals

$$\begin{aligned}
\mathrm{AUC} &= \sum_{j=0}^{n} \sum_{u \subseteq 1:n} g_u(\mathbf{1}) 1_{u \subseteq 1:j} \\
&= \sum_{j=0}^{n} \sum_{u \subseteq 1:n} g_u(\mathbf{1}) 1_{\lceil u \rceil \leqslant j} \\
&= \sum_{u \subseteq 1:n} g_u(\mathbf{1})(n - \lceil u \rceil + 1).
\end{aligned}$$

Finally

$$g_u(\mathbf{1}) = \sum_{v \subseteq u} (-1)^{|u-v|} g(e_v) = \sum_{v \subseteq u} (-1)^{|u-v|} f(\boldsymbol{x}'_v{:}\boldsymbol{x}_{-v}) = \Delta_u f.$$

■

B.1 ABC for Deletion

Now suppose that we use the deletion strategy of replacing x_j by x'_j in the opposite order from that used above, meaning that we change variables thought to most decrease f first. Then letting $\lfloor u \rfloor$ be the index of the smallest element of $u \subseteq 1{:}n$, with $\lfloor \varnothing \rfloor = n + 1$ by convention, we get by the argument in Theorem 3,

$$\mathrm{AUC}' = \sum_{u \subseteq 1:n} \lfloor u \rfloor \Delta_u f.$$

Our area between the curves, ABC, measure for deletion is

$$\begin{aligned}\mathrm{ABC}' &= \frac{n+1}{2}\big(f(\boldsymbol{x})+f(\boldsymbol{x}')\big) - \mathrm{AUC}'\\ &= \sum_{u\neq\varnothing}\Big(\frac{n+1}{2} - \lfloor u\rfloor\Big)\Delta_u f.\end{aligned}$$

If we sum the two ABC measures we get

$$\mathrm{ABC}+\mathrm{ABC}' = \sum_{u\neq\varnothing}(n-\lceil u\rceil-\lfloor u\rfloor+1)\Delta_u f.$$

Proposition 1 gave us a formula for $\mathbb{E}(\lceil\pi(u)\rceil)$ where $\pi(u)$ is the image of the set u under a uniform random permutation of $1{:}n$. By symmetry, we know that

$$\Pr(\lfloor\pi(u)\rfloor=\ell) = \Pr(\lceil\pi(u)\rceil = n-\ell+1)$$

for $0\leqslant\ell\leqslant n+1$. As a result $\mathbb{E}(\lceil u\rceil+\lfloor u\rfloor) = n+1$ from which

$$\mathbb{E}(\mathrm{ABC}+\mathrm{ABC}') = 0.$$

B.2 Monotonicity

Here we prove a sufficient condition under which ranking variables in decreasing order by their Shapley value gives the order that maximizes the insertion AUC. We suppose that $f(\boldsymbol{z}) = h(a(\boldsymbol{z}))$ where $a(\boldsymbol{x})$ is an additive function on $\{0,1\}^n$ and $h:\mathbb{R}\to\mathbb{R}$ is strictly increasing. An additive function on $\boldsymbol{z}\in\{0,1\}^n$ takes the form

$$a(\boldsymbol{z}) = \gamma_0 + \sum_{j=1}^{n}\gamma_j z_j.$$

By choosing $h(w)=\sigma(w)\equiv(1+\exp(-w))^{-1}$ we can study logistic regression probabilities, while $h(w)=w$ accounts for those same probabilities on the logit scale. By choosing $h(w)=\exp(w)$ we can include naive Bayes. Taking $h(w)$ to be the leaky ReLU function we can compare the importance of the inputs to a neuron at some position within a network.

Logistic regression is ordinarily expressed as $\Pr(Y=1\mid\boldsymbol{x}) = \sigma(\beta_0+\boldsymbol{x}^{\mathsf{T}}\beta)$. Then $\Pr(Y=1\mid\boldsymbol{x}') = \sigma(\beta_0+\boldsymbol{x}'^{\mathsf{T}}\beta)$. If we select $\boldsymbol{z}\in\{0,1\}^n$ with $z_j=1$ indicating that we choose x'_j for the j'th component and $j=0$ indicating that we choose x_j for the j'th component, then

$$f(\boldsymbol{z}) = \sigma\Big(\beta_0+\sum_{j=1}^{d}z_j(x'_j-x_j)\beta_j\Big).$$

In other words we take $\gamma_j=(x'_j-x_j)\beta_j$ and $\gamma_0=\beta_0$ to define the function on $\{0,1\}^n$ that we study.

The composite function is $f(\boldsymbol{z})=h(a(\boldsymbol{z}))$. We suppose without loss of generality that $\beta_1\geqslant\beta_2\geqslant\cdots\geqslant\beta_n$. Then the Shapley values satisfy

$$\phi_1\geqslant\phi_2\geqslant\cdots\geqslant\phi_n.$$

To see this, suppose that $u \subseteq 1{:}n$ with $\ell, \ell' \notin u$ and $\ell < \ell'$. Then

$$f(e_{u\cup\ell}) - f(e_{u\cup\ell'}) = h\Big(\beta_0 + \sum_{j\in u}\beta_j + \beta_\ell\Big) - h\Big(\beta_0 + \sum_{j\in u}\beta_j + \beta_{\ell'}\Big) \geqslant 0.$$

It follows that the incremental gains from adding ℓ to any set u not containing ℓ and ℓ' is never smaller than that from adding ℓ' and hence $\phi_\ell \geqslant \phi_{\ell'}$.

Now suppose that we arrange the variables in some order $\pi(j)$ where $\pi(\cdot)$ is a permutation of $1{:}n$. We then get an AUC of

$$\mathrm{AUC}(\pi) = \sum_{j=0}^{n} h\Big(\beta_0 + \sum_{\ell=1}^{j}\beta_{\pi(\ell)}\Big),$$

where the summation over ℓ is zero for $j = 0$. Now let π' be a different permutation that swaps positions r and $r+1$ in π where $1 \leqslant r < n$. It has

$$\begin{aligned}\mathrm{AUC}(\pi') &= \sum_{j=0}^{n} h\Big(\beta_0 + \sum_{\ell=1}^{j}\beta_{\pi'(\ell)}\Big)\\ &= \sum_{j=0}^{n} h\Big(\beta_0 + \sum_{\ell=1}^{j}\big(\mathbf{1}_{\ell<r}\beta_{\pi(\ell)} + \mathbf{1}_{\ell=r}\beta_{\pi(r+1)} + \mathbf{1}_{\ell=r+1}\beta_{\pi(r)} + \mathbf{1}_{\ell>r+1}\beta_{\pi(\ell)}\big)\Big).\end{aligned}$$

Therefore $\mathrm{AUC}(\pi')$ and $\mathrm{AUC}(\pi)$ only differ in the summand for $j = r$ and so

$$\mathrm{AUC}(\pi') - \mathrm{AUC}(\pi) = h\Big(\beta_0 + \sum_{\ell=1}^{r-1}\beta_{\pi(\ell)} + \beta_{\pi(r+1)}\Big) - h\Big(\beta_0 + \sum_{\ell=1}^{r+1}\beta_{\pi(\ell)}\Big).$$

Now if $\beta_{\pi(r+1)} < \beta_{\pi(r)}$ we get $\mathrm{AUC}(\pi') < \mathrm{AUC}(\pi)$. As a result, any maximizer π of AUC must have $\beta_{\pi(r+1)} \geqslant \beta_{\pi(r)}$ for all $r = 1, \ldots, n-1$.

Next we consider integrated gradients for this setting, assuming that h is differentiable with $h' > 0$. The gradient is then $h'(\boldsymbol{z})\beta$. The gradient at any point then sorts the inputs in the same order as the Shapley value. Therefore any positive linear combination of those gradient evaluations sorts the inputs into this order which then optimizes the deletion AUC.