# Feature Importance for Model Fit: Nonlinear Regression and Classification in Machine Learning Models

Ludger Hentschel

May 17, 2026

**Abstract**

We study the problem of attributing predictive performance to input features in fixed, fitted machine learning models. We define feature importance as contribution to explained predictive fit, measured as the reduction in expected loss relative to a baseline prediction, and call the resulting framework Euler Decomposition of Explained Fit (EDEF). This formulation applies uniformly across regression and classification and focuses on realized predictive performance rather than individual predictions or counterfactual model variants.

EDEF derives an exact, additive decomposition of explained fit by applying the fundamental theorem of calculus along a path in input space. The reduction in loss is expressed as a sum of feature-level contributions defined by integrated loss gradients. The attribution is global, additive, and model-conditional, and generalizes Euler-style decompositions to nonlinear machine learning models, including neural networks, tree-based methods, additive models, and kernel methods. When automatic differentiation is available, computation is efficient and scales well to high-dimensional settings.

We also derive standard errors for the EDEF contributions based on observation-level variation, enabling statistical comparison of feature importance across samples, time periods, or model variants.

www.ludgerhentschel.com ludger@ludgerhentschel.com

# Contents

**Acknowledgements**
For helpful comments, I am grateful to Nishant Gurnani and Shubham Jaiswal.

# 1 Introduction

Modern machine learning models are often evaluated by their predictive performance, as discussed in standard treatments such as Hastie, Tibshirani, and Friedman (2009) and Vapnik (1998), but performance alone does not explain how that performance is generated. Once a model has been trained and deployed, it is natural to ask which input features contribute to its realized predictive accuracy, whether those contributions are stable across samples or over time, and whether features that strongly affect individual predictions also improve overall predictive fit. These questions are related but distinct, and the last is the subject of this paper.

This paper develops a framework for attributing predictive fit to input features in fixed, fitted machine learning models. We define feature importance as contribution to model fit: the contribution of an input variable to the reduction in loss relative to a fixed baseline predictor, and we call the resulting framework Euler Decomposition of Explained Fit (EDEF). EDEF is global, additive, and model-conditional. It provides an exact decomposition of predictive performance with no residual terms and, uniquely among methods that address model-fit attribution directly, delivers standard errors for each feature contribution.

EDEF applies to both regression and classification. In regression, predictive fit may be measured by the reduction in squared error loss or by another differentiable regression loss. In classification, predictive fit may be measured by the reduction in cross-entropy, log loss, or another differentiable classification loss relative to a baseline probability model. In all cases, the object of attribution is a scalar measure of predictive performance, not an individual prediction, probability, score, or fitted value.

Let $x \in \mathbb{R}^K$ denote a vector of input features, let $y$ denote the corresponding outcome or label, and let $f(\widehat{\theta}, \cdot)$ denote the prediction function produced by a trained model. The prediction may be a scalar fitted value, a probability, a vector of class probabilities, or a score that is transformed into such quantities before the loss is evaluated. We treat the fitted parameters $\widehat{\theta}$ as fixed and require only that the fitted prediction function can be evaluated at admissible inputs. The evaluation sample used for attribution may differ from the training sample used to fit the model.

We measure predictive fit as the reduction in expected loss relative to a fixed baseline prediction. To attribute this reduction to input variables, we trace a path from a baseline input $x_0$ to the realized input $x$ and evaluate how the loss changes along that path. The resulting construction follows directly from the fundamental theorem of calculus and yields an exact additive

decomposition of the change in loss. With a straight-line path,

$$x(t) = x_0 + t(x - x_0), \quad t \in [0, 1], \tag{1}$$

the contribution of feature $j$ is obtained by integrating the loss gradient with respect to $x_j$ along the path and then averaging across observations. Positive contributions correspond to reductions in loss and therefore to improvements in predictive fit.

The resulting EDEF attribution generalizes Euler-style decompositions of explained fit beyond linear models. In the context of risk and performance attribution, Tasche (2008) develops Euler allocations for homogeneous functionals, and Hentschel (2026a) shows how such ideas apply to linear models. In linear regression and logistic regression, the path-integral construction collapses to closed-form Euler decompositions. In nonlinear models, the same logic remains valid, but the attribution must account for nonlinearities and interactions by integrating marginal loss contributions along the path from baseline inputs to realized inputs. The path provides an explicit convention for allocating interaction effects across features.

EDEF is designed for modern machine learning models. It applies to prediction functions that are differentiable almost everywhere along the chosen path, including neural networks, generalized additive models, spline-based models, and kernel methods of the type studied by Schölkopf and Smola (2002). It also accommodates piecewise-smooth and piecewise-constant methods, including tree-based models and their ensembles, as introduced by Breiman (2001) and extended by Friedman (2001). EDEF does not require access to the estimation procedure, does not require refitting, and does not require evaluating counterfactual feature subsets.

The computational requirements are predictable. When gradients with respect to inputs are available through automatic differentiation, as developed by Griewank and Walther (2008) and surveyed by Baydin, Pearlmutter, Radul, and Siskind (2018), each quadrature node requires a forward and backward evaluation of the fitted model, producing the full input gradient simultaneously. When automatic differentiation is not available, finite differences provide a model-agnostic fallback. In either case, the numerical integration is one-dimensional along the path from $x_0$ to $x$, not an integration over the full feature space. For tree-based models, the path integral reduces exactly to a weighted sum over split-crossing events, yielding further computational gains. In practice, EDEF is orders of magnitude faster than SAGE, the closest existing method for model-fit attribution, and competitive with or faster than

prediction-explanation methods that were not designed to answer questions about model fit.

In addition to feature contributions, we derive standard errors for the EDEF attributions. These standard errors are computed from observation-level contributions and reflect sampling variability in the evaluation data, conditional on the fitted model. No existing method for model-fit attribution provides standard errors.[1] The EDEF standard errors allow feature contributions to be compared across samples, model versions, time periods, or subpopulations in statistical terms, and support formal inference about which features drive predictive performance.

There is a natural temptation to use prediction-explanation methods such as SHAP or integrated gradients to assess which features drive model fit. These methods are widely used, well-understood, and computationally feasible. But they are designed to answer a different question — why does a fixed, fitted model produce a particular prediction? — and there is no reason to expect their output to reliably reflect contribution to predictive fit. A feature may have large SHAP values because it strongly influences predictions, while contributing little to fit if those predictions are not aligned with the outcome. Conversely, a feature may contribute substantially to fit while having modest local effects on any individual prediction. The divergence is not merely conceptual. In our classification example, the rank correlation between EDEF and SHAP feature importance stands at 0.86 when the model is evaluated on clean data, but falls to near zero as predictive fit deteriorates, even though the fitted model is unchanged. This sensitivity of SHAP to the alignment between predictions and outcomes, which SHAP does not directly measure, illustrates why prediction-explanation methods are not substitutes for fit attribution.

Methods based on cooperative game theory, beginning with Shapley (1953) and implemented for machine learning models by Lundberg and Lee (2017), are widely used to attribute predictions to features. The integrated gradients method of Sundararajan, Taly, and Yan (2017) uses path integration to explain individual predictions. Classical regression diagnostics, such as those developed by Belsley, Kuh, and Welsch (1980), measure local sensitivity, while permutation-based approaches following Breiman (2001) and Fisher, Rudin, and Dominici (2019) assess feature reliance. More recent work by Covert, Lundberg, and Lee (2020) and Covert, Lundberg, and Lee (2021) extends Shapley-style ideas to model performance by considering feature removal. These are valuable contributions to the literature on model understanding, but

[1] Some other measures report approximation accuracy, which does not measure sample variation in a conventional statistical sense.

none of them directly decomposes realized predictive performance of a fixed, fitted model. Within this narrower and well-defined question, EDEF provides an exact, computationally efficient, and inferentially complete solution.

The remainder of the paper proceeds as follows. Section 2 defines predictive fit and formalizes feature importance as contribution to explained fit. Section 3 derives the general path-integral attribution for scalar losses. Section 4 specializes the framework to regression and classification losses. Section 5 discusses implementation for major machine learning model classes. Section 6 derives standard errors and grouped contributions. Section 7 compares EDEF to other feature-importance methods. Section 8 presents nonlinear regression and classification examples, and Section 9 concludes.

## 2 Contributions to Predictive Fit

Our objective is to attribute predictive performance to input features in a fixed, fitted model. To do so, we first define the object of interest and then formalize our notion of feature importance as contribution to that object.

### 2.1 Predictive Fit

Let $y$ denote the observed outcome or label and let $\widehat{y}(x)$ denote the prediction produced by a fitted model for input $x \in \mathbb{R}^K$. The prediction may be a scalar fitted value, a probability, a vector of class probabilities, or a score that is mapped into such quantities prior to evaluation. Let $\ell(y, \widehat{y})$ denote a scalar loss function that is applied to each observation.

We evaluate predictive performance through the expected loss

$$\mathcal{L}(\widehat{y}) = \mathbb{E}[\ell(y, \widehat{y})], \tag{2}$$

where the expectation is taken over an evaluation sample. This sample may coincide with the training data or may be out-of-sample.

To define explained predictive fit, we introduce a baseline prediction $\widehat{y}_0$ that represents an uninformed or minimally informed model. Typical choices include a constant mean for regression or class frequencies for classification, but more structured baselines are admissible provided they remain fixed throughout the attribution exercise.

We define explained fit as the reduction in expected loss relative to this baseline,

$$\Delta\mathcal{L} = \mathcal{L}(\widehat{y}_0) - \mathcal{L}(\widehat{y}). \tag{3}$$

A larger value of $\Delta\mathcal{L}$ corresponds to better predictive performance.

This formulation applies uniformly across regression and classification. In regression, $\ell$ may be squared error or another differentiable loss. In classification, $\ell$ may be cross-entropy, log loss, or another differentiable classification loss. In all cases, the object of interest is a scalar measure of predictive performance.

## 2.2 Feature Importance

We define feature importance as the contribution of an input variable to the explained predictive fit $\Delta\mathcal{L}$ within a fixed fitted model.

This notion of importance has three key properties.

First, it is *global*. It attributes overall predictive performance rather than explaining individual predictions. The object being decomposed is the aggregate reduction in loss across observations.

Second, it is *additive*. We seek a decomposition of the form

$$\Delta\mathcal{L} = \sum_{j=1}^{K} C_j, \tag{4}$$

where $C_j$ is the contribution of feature $x_j$. Additivity ensures that the full predictive performance of the model is allocated across features without residual terms.

Third, it is *model-conditional*. The attribution is computed for a fixed fitted model and a fixed baseline. It describes how predictive performance is generated by the model actually used, not how performance would change under feature removal, perturbation, or refitting. Removing or modifying a feature generally produces a different model and therefore a different attribution.

This definition differs from several common notions of feature importance. Measures based on refitting or feature removal assess feature reliance or substitutability across models. Gradient-based and Shapley-based methods are often used to explain individual predictions. Perturbation and permutation methods measure sensitivity of predictions or loss to input disruption. These approaches address different questions and need not agree with contribution to predictive fit.

## 2.3 Interpretation

Under this framework, a feature is important if it contributes materially to reducing predictive loss relative to the baseline. Contributions $C_j$ may be positive or negative. A positive contribution indicates that the feature helps reduce prediction error on average, while a negative contribution

indicates that, conditional on the fitted model, the feature degrades predictive performance.

Negative contributions do not imply that removing the feature would improve performance. Because the attribution is model-conditional and additive, contributions are jointly determined. Changing one feature typically changes the contributions of all others. The decomposition provides a diagnostic of how predictive performance is generated within the fitted model, rather than a prescription for feature selection.

Finally, because the object of interest is a scalar measure of predictive fit, the resulting attribution is directly comparable across model classes, loss functions, and applications. This allows feature contributions to be analyzed consistently in regression and classification settings and across a wide range of machine learning models.

## 3 Path-Integral Attribution for Scalar Losses

We now derive an additive decomposition of explained predictive fit for a fixed fitted model. The construction applies to any differentiable scalar loss and covers both regression and classification settings within a unified framework.

### 3.1 Setup

Let $y \in \mathbb{R}^N$ denote a vector of observed outcomes or labels, and let $X \in \mathbb{R}^{N \times K}$ denote the corresponding matrix of input features. Each row $x_i \in \mathbb{R}^K$ represents the feature vector for observation $i$.

Let $\widehat{y}(x)$ denote the prediction produced by a fitted model for a single input $x$. The prediction may be a scalar fitted value, a probability, or a vector that is mapped into a scalar loss. For notational simplicity, we treat $\widehat{y}(x)$ as the argument entering the loss function.

We evaluate predictive performance using a differentiable loss function $\ell(y_i, \widehat{y}_i)$ applied element-wise across observations. Writing

$$\mathcal{L}(\widehat{y}) = \mathbb{E}[\ell(y, \widehat{y})], \tag{5}$$

we define explained fit as

$$\Delta\mathcal{L} = \mathcal{L}(\widehat{y}_0) - \mathcal{L}(\widehat{y}(X)), \tag{6}$$

where $\widehat{y}_0$ is a fixed baseline prediction.

To attribute $\Delta\mathcal{L}$ to input features, we introduce a baseline input vector $x_0 \in \mathbb{R}^K$ and compare predictions at $x_0$ and at the realized input $x$. Under

standard centering conventions, $x_0 = 0$ is a natural choice, but alternative baselines are admissible.

## 3.2 Path-Integral Decomposition

Our starting point is an Euler-style objective: to decompose a scalar measure of predictive fit into additive contributions across input features. In linear models, such decompositions follow directly from Euler's theorem for homogeneous functions, as discussed by Silberberg (1978) and Tasche (2008).

In nonlinear models, explained fit is not a homogeneous function of the inputs, and a direct Euler decomposition is no longer available. To extend Euler attribution to this setting, we instead apply the fundamental theorem of calculus along a path in input space. This yields an exact additive decomposition of changes in loss without requiring homogeneity.

For each observation, we consider a smooth path in input space connecting the baseline input $x_0$ to the realized input $x$. We adopt the straight-line path

$$x(t) = x_0 + t(x - x_0), \quad t \in [0, 1], \tag{7}$$

which treats all features symmetrically and introduces no additional structure. This is the same path used by Sundararajan, Taly, and Yan (2017), although in that work the path integral is applied to the prediction function to explain individual predictions, whereas we apply it to the loss function to decompose predictive fit.

Applying the fundamental theorem of calculus to the composite function $t \mapsto \ell(y, f(x(t)))$ yields

$$\ell(y, \widehat{y}(x)) - \ell(y, \widehat{y}(x_0)) = \int_0^1 \frac{d}{dt} \ell(y, \widehat{y}(x(t)))\, dt \tag{8}$$

$$= \int_0^1 (x - x_0)^\top \nabla_x \ell(y, \widehat{y}(x(t)))\, dt. \tag{9}$$

Rewriting the inner product produces an additive decomposition across input coordinates,

$$\ell(y, \widehat{y}(x)) - \ell(y, \widehat{y}(x_0)) = \sum_{j=1}^{K} (x_j - x_{0j}) \int_0^1 \frac{\partial}{\partial x_j} \ell(y, \widehat{y}(x(t)))\, dt. \tag{10}$$

This identity holds for each observation. Averaging across observations

yields a global decomposition of explained fit,

$$\Delta\mathcal{L} = \sum_{j=1}^{K} C_j, \tag{11}$$

where the contribution of feature $x_j$ is defined as

$$C_j = -\mathbb{E}\left[(x_j - x_{0j}) \int_0^1 \frac{\partial}{\partial x_j} \ell(y, \widehat{y}(x(t)))\, dt\right]. \tag{12}$$

The leading minus sign reflects the fact that explained fit is defined as a reduction in loss. Positive contributions correspond to features that improve predictive performance. By construction, the contributions $C_j$ sum exactly to the total reduction in expected loss.

Hentschel (2026a) shows that for linear models the contributions in equation (12) admit closed-form solutions. The nonlinear models that are the focus here generally require numerical integration.

Since EDEF provides exact additive attribution, we can confirm the numerical accuracy of the integration. This allows us to perform numerical integration over a fairly crude initial grid. We only need to incur the higher computational burden of a finer grid if the initial calculations are not sufficiently accurate.

### 3.3 Extension to Piecewise-Constant Models

The derivation above assumes that $x \mapsto \ell(y, \widehat{y}(x))$ is differentiable along the interpolation path. Tree-based models produce piecewise-constant prediction functions and therefore are not differentiable in the ordinary sense: prediction gradients vanish between split boundaries and are undefined at split boundaries. Nevertheless, the path-integral decomposition extends naturally to such models through a generalized derivative interpretation.

Along the straight-line path

$$x(t) = x_0 + t(x - x_0), \tag{13}$$

the prediction of a tree model changes only when the path crosses a split boundary. The path integral therefore reduces exactly to a finite sum over split-boundary crossings. If the path crosses boundaries at times

$$0 < t_1 < t_2 < \cdots < t_Q < 1, \tag{14}$$

with prediction jumps

$$\Delta\widehat{y}_q = \widehat{y}_q^{+} - \widehat{y}_q^{-}, \tag{15}$$

then the total change in loss decomposes as

$$\ell(y, \widehat{y}(x)) - \ell(y, \widehat{y}(x_0)) = \sum_{q=1}^{Q} \Big( \ell(y, \widehat{y}_q^{+}) - \ell(y, \widehat{y}_q^{-}) \Big), \tag{16}$$

where $\widehat{y}_q^{-}$ and $\widehat{y}_q^{+}$ denote the prediction values immediately before and after the $q$th crossing.

Each crossing is attributed to the feature defining the corresponding split. For ensembles, contributions are summed across all trees and all crossings along the path. The resulting decomposition remains exact and additive and avoids numerical quadrature entirely.

This crossing-sum representation is the nonsmooth analogue of the smooth path-integral decomposition derived above. Hentschel (2026b) provides a detailed treatment of the generalized derivative interpretation, distributional formulation, and efficient crossing-sum algorithms.

## 3.4 Interpretation

The contribution $C_j$ aggregates the marginal effect of feature $x_j$ on the loss as the input moves from the baseline $x_0$ to the realized value $x$. Each term combines three elements: variation in the input $(x_j - x_{0j})$, local sensitivity of the loss with respect to that input, and the evolution of these quantities along the path.

Because the attribution integrates over a continuous path in input space, it captures nonlinearities and interactions among features. Interaction effects enter through the loss gradient and are allocated across features according to the chosen path. In linear models, this allocation is immaterial, but in nonlinear models it reflects an explicit convention.

Contributions need not be positive. A negative value of $C_j$ indicates that, conditional on the fitted model, variation in feature $x_j$ is associated with increases in loss along the path and therefore reduces predictive fit on average.

## 3.5 Regularity Conditions

The path-integral construction requires that the mapping $x \mapsto \ell(y, \widehat{y}(x))$ be differentiable almost everywhere along the path from $x_0$ to $x$, or that any points of non-differentiability admit a generalized derivative interpretation of the

type established in section 3.3. This condition is satisfied for all model classes considered in this paper: smooth models including neural networks with ReLU activations and kernel methods, piecewise-smooth models including splines and generalized additive models, and piecewise-constant models including decision trees and their ensembles.

## 3.6 Path Dependence

The total change in loss between $x_0$ and $x$ is path independent, as implied by the fundamental theorem of line integrals, but the allocation of that change across features is not. Different paths correspond to different conventions for assigning interaction effects.

The straight-line path adopted here traverses the interior of the feature space and treats all inputs symmetrically. When interactions are weak or the model is approximately additive, different smooth paths yield similar attributions. When interactions are strong, path dependence becomes economically meaningful and reflects how joint effects are distributed across features.

The need to specify a path is not a limitation of the method but a direct consequence of nonlinearity.

## 3.7 Relation to Attribution Axioms

Although we derive EDEF directly from the decomposition of predictive fit rather than from an axiomatic characterization, it satisfies several standard properties discussed in the attribution literature.

The EDEF decomposition satisfies:

1. *Completeness*: The contributions sum exactly to realized predictive fit;
2. *Dummy feature*: Features with zero pathwise gradient contribution receive zero attribution;
3. *Symmetry*: Features entering the prediction function symmetrically receive equal attribution under symmetric paths;
4. *Implementation invariance*: The decomposition depends only on the realized prediction function and loss, not on the internal parameterization or representation of the model.

These properties follow directly from the path-integral representation. Completeness holds by construction; dummy feature and symmetry follow from the structure of the integrand; implementation invariance follows from the fact that the integral depends only on the input-output mapping, not on how that mapping is internally computed.

# 4 Regression and Classification Losses

The path-integral attribution developed in the previous section applies to any differentiable scalar loss. We now specialize the framework to the loss functions most commonly used in regression and classification. This specialization clarifies the interpretation of the feature contributions and connects the general construction to familiar performance measures.

## 4.1 Regression Losses

The standard measure of predictive fit in regression is squared error loss, as discussed by Hastie, Tibshirani, and Friedman (2009),

$$\ell(y_i, \widehat{y}_i) = (y_i - \widehat{y}_i)^2. \tag{17}$$

The derivative with respect to the prediction is

$$\frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \widehat{y}_i} = -2(y_i - \widehat{y}_i). \tag{18}$$

Applying the chain rule yields the gradient of the loss with respect to the inputs,

$$\frac{\partial}{\partial x_j} \ell(y, f(x)) = -2(y - f(x)) \frac{\partial f(x)}{\partial x_j}. \tag{19}$$

Substituting into the general definition of the feature contributions gives

$$C_j = 2\,\mathbb{E}\left[ (x_j - x_{0j}) \int_0^1 \big(y - f(x(t))\big) \frac{\partial f(x(t))}{\partial x_j}\, dt \right]. \tag{20}$$

Under squared error loss, each contribution combines three elements: the variation in the input, $(x_j - x_{0j})$, the sensitivity of the fitted signal to that input, $\partial f(x(t))/\partial x_j$, and the residual along the path from the baseline input to the realized input, $(y - f(x(t))$. Features generate larger positive contributions when their variation aligns with directions in input space along which the fitted model both changes strongly and reduces prediction error.[2]

[2] More generally, we can extend the framework to any differentiable regression loss, such as absolute error, Huber loss, or other smooth alternatives. In each case, the derivative $\partial \ell / \partial \widehat{y}$ rescales the local signal sensitivity according to how changes in the prediction affect the loss. The interpretation remains the same: contributions measure alignment between input variation and reductions in loss along the path.

Hentschel (2026a) shows that, in the case of linear regressions, the contributions are

$$C_j = 2\,\mathrm{Cov}(y, \widehat{y}_j) - \mathrm{Cov}(\widehat{y}, \widehat{y}_j) \tag{21}$$

$$= \mathrm{Cov}(y, \widehat{y}_j) + \mathrm{Cov}(e, \widehat{y}_j). \tag{22}$$

We assign a high contribution to feature $j$ if its share of the prediction, $\widehat{y}_j$, aligns well with the actual outcome $y$ and if it moves the overall prediction toward the outcome. In nonlinear models, no such global covariance decomposition exists, because both the sensitivity of the fitted model and the residual depend on the input location.

## 4.2 Binary Classification

In binary classification, the model produces a probability $p(x) \in (0, 1)$ for the event $y = 1$. Predictive performance is commonly evaluated using log loss or cross-entropy loss, as in Hastie, Tibshirani, and Friedman (2009),

$$\ell(y, p) = -y \log p - (1 - y) \log(1 - p). \tag{23}$$

Log loss is a strictly proper scoring rule, meaning that its expected value is uniquely minimized by the true conditional probability, as shown by Gneiting and Raftery (2007). This property makes it a natural objective for evaluating probabilistic predictions and for attributing predictive fit.[3]

The derivative of log loss with respect to the predicted probability is

$$\frac{\partial \ell(y, p)}{\partial p} = -\frac{y}{p} + \frac{1 - y}{1 - p}. \tag{24}$$

Applying the chain rule,

$$\frac{\partial}{\partial x_j} \ell(y, p(x)) = \frac{\partial \ell(y, p)}{\partial p} \frac{\partial p(x)}{\partial x_j}. \tag{25}$$

Substituting into the general definition yields

$$C_j = -\mathbb{E}\left[ (x_j - x_{0j}) \int_0^1 \frac{\partial \ell(y, p(x(t)))}{\partial p} \frac{\partial p(x(t))}{\partial x_j}\, dt \right]. \tag{26}$$

[3] By contrast, commonly used metrics such as classification accuracy, $F_1$ score, or AUC are not strictly proper scoring rules. They also are generally not differentiable in the model outputs. As a result, they do not naturally admit a path-integral decomposition of the type developed here and are not well suited for additive attribution of predictive performance.

It is often convenient to express the model in terms of a score $s(x) \in \mathbb{R}$ with $p(x) = \sigma(s(x))$, where $\sigma$ is the logistic function, a standard representation for probabilistic classifiers described by Hastie, Tibshirani, and Friedman (2009). In this case, the derivative simplifies to

$$\frac{\partial \ell(y, p(x))}{\partial s} = p(x) - y, \tag{27}$$

and the input gradient becomes

$$\frac{\partial}{\partial x_j} \ell(y, p(x)) = (p(x) - y) \frac{\partial s(x)}{\partial x_j}. \tag{28}$$

Under log loss, the contribution of feature $x_j$ reflects how variation in that feature aligns with reductions in classification error as measured by the log-likelihood. The term $(p(x) - y)$ plays the role of a residual, measuring the discrepancy between predicted probabilities and realized outcomes under log loss.

Hentschel (2026a) shows that linear classification models produce a closed-form solution for the contributions

$$C_j = \mathbb{E}\left[\widehat{\beta}_j x_j \left(y - \frac{\log(1 + e^{\widehat{\eta}}) - \log(1 + e^{\overline{\eta}})}{\widehat{\eta} - \overline{\eta}}\right)\right] \tag{29}$$

$$\widehat{\eta} = \widehat{\beta}_0 + \sum_{j=1}^{K} \widehat{\beta}_j x_j. \tag{30}$$

## 4.3 Multiclass Classification

In multiclass classification with $G$ classes, the model produces a vector of probabilities $p(x) = (p_1(x), \ldots, p_G(x))$ with $\sum_g p_g(x) = 1$. The standard loss is cross-entropy,

$$\ell(y, p) = -\sum_{g=1}^{G} \mathbf{1}\{y = g\} \log p_g(x). \tag{31}$$

Let $s_g(x)$ denote the score for class $g$, with probabilities given by the softmax transformation,

$$p_g(x) = \frac{\exp(s_g(x))}{\sum_h \exp(s_h(x))}. \tag{32}$$

The derivative of the loss with respect to the scores is

$$\frac{\partial \ell(y, p(x))}{\partial s_g} = p_g(x) - \mathbf{1}\{y = g\}. \tag{33}$$

Applying the chain rule,

$$\frac{\partial}{\partial x_j} \ell(y, p(x)) = \sum_{g=1}^{G} \big(p_g(x) - \mathbf{1}\{y = g\}\big) \frac{\partial s_g(x)}{\partial x_j}. \tag{34}$$

Substituting into the path-integral formula yields feature contributions that aggregate the effect of each input across all classes.

As in the binary case, the terms $(p_g(x) - \mathbf{1}\{y = g\})$ act as class-specific residuals. The contribution of feature $x_j$ reflects how its variation aligns with improvements in classification performance across all classes.

### 4.4 Interpretation Across Losses

Across regression and classification, the structure of the attribution is the same. Feature contributions combine variation in inputs, sensitivity of the model with respect to those inputs, and a loss-dependent residual that captures how prediction errors translate into changes in performance.

This structure ensures that the attribution remains comparable across model classes and applications. Differences in loss functions affect how prediction errors are weighted, but not the underlying logic of the decomposition. In all cases, contributions measure how input variation aligns with reductions in a well-defined scalar loss along the path from baseline inputs to realized inputs.

## 5 Application to Major ML Model Classes

The path-integral attribution applies broadly to modern machine learning models. The key requirement is that the composite mapping $x \mapsto \ell(y, \widehat{y}(x))$ be differentiable almost everywhere along the path from the baseline input $x_0$ to the realized input $x$. This section makes explicit how the framework applies to the major classes of models used in practice and how the required gradients can be computed.

A central advantage of the approach is that it operates directly on the fitted prediction function $\widehat{y}(x)$. It does not depend on the estimation procedure, does not require refitting, and does not rely on access to internal parameters beyond what is needed to evaluate predictions and their gradients with respect to inputs.

## 5.1 Neural Networks

Neural networks, developed in modern form through backpropagation by Rumelhart, Hinton, and Williams (1986) and surveyed by Hastie, Tibshirani, and Friedman (2009), are a natural setting for the path-integral attribution. Feedforward networks, convolutional networks, and many sequence models define prediction functions that are differentiable almost everywhere in the inputs.

Modern machine learning frameworks expose the computational graph of the fitted model and support automatic differentiation, as described by Griewank and Walther (2008). Using reverse-mode automatic differentiation, we can compute the full gradient $\nabla_x \ell(y, \widehat{y}(x))$ with respect to all input features in a single backward pass. As a result, each quadrature node along the path requires one forward evaluation of the model and one backward pass to obtain all feature gradients simultaneously.

Automatic differentiation yields a computational cost that is essentially independent of the number of input features. In this case, the attribution scales efficiently to high-dimensional inputs and deep architectures.

For networks with piecewise-linear activation functions such as rectified linear units, the prediction function is not differentiable everywhere. These points of non-differentiability occur on sets of measure zero and do not affect the value of the path integral.

## 5.2 Tree-Based Models

Tree-based models, including decision trees (Breiman et al., 1984), random forests (Breiman, 2001), and gradient boosting methods (Friedman, 2001), produce piecewise- constant prediction functions. As discussed in section 3.3, the path-integral decomposition extends naturally to these nonsmooth models through a generalized derivative interpretation: along the interpolation path, contributions arise only when the path crosses a split boundary, so the path integral reduces exactly to a finite crossing sum.

For axis-aligned trees, each crossing is attributed to the feature defining the corresponding split. In ensembles, contributions are summed across all trees and all crossings along the path. The resulting decomposition remains exact and additive and avoids numerical quadrature entirely. For an ensemble of $T$ trees of depth $d$, the computational cost is approximately $O(Td)$ per observation, plus a sorting step required to order crossings along the path.

This decomposition is model-conditional and attributes realized predictive fit rather than predictions themselves. It therefore differs conceptually from prediction-attribution methods such as TreeSHAP (Lundberg and Lee, 2017), which decompose predictions rather than predictive performance.

A detailed treatment of the generalized derivative representation, the crossing-sum formulation, and efficient algorithms for tree-path integration appears in Hentschel (2026b).

## 5.3 Generalized Additive Models and Spline Methods

Generalized additive models were developed by Hastie and Tibshirani (1990) and provide a bridge between linear and highly nonlinear models. In these models, the prediction function takes the form

$$\widehat{y}(x) = \widehat{\alpha} + \sum_{j=1}^{K} g_j(x_j), \tag{35}$$

possibly with additional low-dimensional interaction terms.

Spline-based methods, as discussed by Wahba (1990), provide flexible implementations of the component functions $g_j$.

Because the model is additive or nearly additive, the path-integral attribution exhibits limited path dependence. Contributions for each feature are primarily determined by the corresponding component function $g_j(x_j)$, and interaction effects are either absent or explicitly modeled.

These models provide a useful benchmark for interpreting the attribution in more complex nonlinear settings, where interactions are stronger and path dependence becomes more pronounced.

## 5.4 Kernel Methods

Kernel methods, including support vector machines introduced by Cortes and Vapnik (1995) and further developed by Vapnik (1998), and kernel ridge regression as discussed by Schölkopf and Smola (2002), define prediction functions of the form

$$\widehat{y}(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i), \tag{36}$$

where $K(\cdot, \cdot)$ is a kernel function.

For commonly used kernels, such as the Gaussian or polynomial kernel, the prediction function is smooth in the inputs. Gradients with respect to inputs can be computed analytically or via automatic differentiation when the model is implemented in a differentiable framework.

The path-integral attribution therefore applies directly, with feature contributions reflecting how variation in each input dimension affects the kernel-based prediction and, through the loss, predictive performance.

### 5.5 State-Space and Sequence Models

State-space models were developed in control theory by Kalman (1960) and are widely used in econometrics and time-series analysis, as discussed by Hamilton (1994), while recurrent neural networks provide a modern machine learning analogue.

Many models for time series and sequential data produce predictions that depend on current and past inputs. When these models define a differentiable mapping from the current input vector to the prediction, the path-integral attribution can be applied in the same way as for static models.

In such settings, the input vector $x$ may include lagged variables, state estimates, or other derived features. The attribution then measures the contribution of these inputs to predictive fit, conditional on the model's internal dynamics.

## 6 Standard Errors and Grouped Contributions

The path-integral attribution decomposes explained predictive fit into feature-level contributions by averaging observation-level quantities across the evaluation sample. This structure allows standard errors to be computed directly from the cross-sectional variation in these observation-level contributions. The resulting inference applies uniformly across regression and classification models and across the range of machine learning models discussed above.

### 6.1 Observation-Level Contributions

Recall that the global contribution of feature $x_j$ is defined as

$$C_j = -\mathbb{E}\left[(x_j - x_{0j}) \int_0^1 \frac{\partial}{\partial x_j} \ell(y, \widehat{y}(x(t)))\, dt\right]. \tag{37}$$

For each observation $i$, define the corresponding observation-level contribution

$$c_{ij} = -(x_{ij} - x_{0j}) \int_0^1 \frac{\partial}{\partial x_j} \ell(y_i, \widehat{y}(x_i(t)))\, dt. \tag{38}$$

By construction, the global contribution is the sample average

$$C_j = \mathbb{E}[c_{ij}] = \frac{1}{N} \sum_{i=1}^{N} c_{ij}. \tag{39}$$

This representation holds regardless of the model class or the specific loss function, provided the loss is differentiable almost everywhere along the

path. In practice, the integral over $t$ is evaluated numerically using a fixed quadrature rule.

## 6.2 Standard Errors

Treating the fitted model as fixed, the variability in $C_j$ arises from sampling variation across observations. Under standard regularity conditions for sample averages of i.i.d. observations, a central limit theorem applies, as discussed by Wooldridge (2002),

$$\sqrt{N}(C_j - \mathbb{E}[c_{ij}]) \xrightarrow{d} \mathcal{N}\left(0, \mathbb{E}[(c_{ij} - C_j)^2]\right). \tag{40}$$

An estimator of the standard error of $C_j$ is

$$SE(C_j) = \sqrt{\frac{1}{N}\mathbb{E}\left[(c_{ij} - C_j)^2\right]}. \tag{41}$$

In empirical applications, this quantity is estimated using the sample variance of the observation-level contributions,

$$\widehat{SE}(C_j) = \sqrt{\frac{1}{N(N-1)}\sum_{i=1}^{N}(c_{ij} - C_j)^2}. \tag{42}$$

These standard errors quantify sampling variability in the evaluation data conditional on the fitted model. They do not incorporate uncertainty from the estimation of model parameters. This is appropriate in settings where the model is treated as fixed, such as performance monitoring, model comparison, or analysis of deployed systems.

The same logic applies across regression and classification losses. The form of the loss affects the magnitude and distribution of the $c_{ij}$, but not the structure of the standard error estimator.

## 6.3 Grouped Contributions

In many applications, it is useful to aggregate features into groups, either for interpretability or to stabilize noisy individual contributions. Because the decomposition is additive, grouped contributions are obtained by summation. This aggregation parallels the group-attribution logic of Owen (1977), but arises here without the need for combinatorial averaging or counterfactual evaluation.

Let $G \subset \{1, \ldots, K\}$ denote a set of feature indices. Define the grouped contribution as

$$C_G = \sum_{j \in G} C_j. \tag{43}$$

The corresponding observation-level grouped contribution is

$$c_{iG} = \sum_{j \in G} c_{ij}, \tag{44}$$

so that

$$C_G = \mathbb{E}[c_{iG}] = \frac{1}{N} \sum_{i=1}^{N} c_{iG}. \tag{45}$$

The standard error of the grouped contribution is obtained directly from the sample variance of $c_{iG}$,

$$\widehat{SE}(C_G) = \sqrt{\frac{1}{N(N-1)} \sum_{i=1}^{N} (c_{iG} - C_G)^2}. \tag{46}$$

This computation automatically accounts for covariance among features within the group and avoids constructing or storing a full covariance matrix of the individual contributions. This is particularly advantageous in high-dimensional settings.

In the empirical examples, we show that grouping allows us to expand categorical features into an appropriate number of one-hot dummy variables, compute the contribution for each dummy variable, and then aggregate the contributions back to the original feature.

## 6.4 Interpretation and Scope

The standard errors derived here apply uniformly across model classes, including neural networks, tree-based models, kernel methods, and other nonlinear architectures. The only requirement is that the observation-level contributions $c_{ij}$ can be computed, which in turn requires evaluating the model and its input sensitivities along the chosen path.

Because the attribution is model-conditional, the standard errors measure variation in realized contributions rather than uncertainty about the model itself. They are most useful for comparing feature contributions across

samples, time periods, or model versions, and for assessing whether observed differences are statistically meaningful.

The observation-level formulation makes it straightforward to compute confidence intervals, perform hypothesis tests, and analyze grouped contributions without additional computational burden once the path-integral contributions have been evaluated.

## 7 Relation to Other Feature Importance Methods

Feature importance measures are designed to answer different questions. We distinguish three types.

The first concerns *model exploration*: which features are informative, and how does model performance change when features are added or removed? Methods such as partial $R^2$, dominance analysis, and related variance-decomposition approaches address this question by comparing predictive performance across alternative specifications.

The second concerns *prediction explanation*: why does a fixed, fitted model produce a particular prediction? Methods such as SHAP and integrated gradients attribute predictions to inputs relative to a baseline. These methods explain predictions, not predictive accuracy.

The third question concerns *model-fit attribution*: how much does each feature contribute to the realized predictive accuracy of a fixed, fitted model? This question is model-conditional and evaluates the model only at the realized inputs. This is a key question in model evaluation, validation, and monitoring.

These questions are related but distinct, and methods designed for one generally do not provide valid answers to the others. The framework developed here addresses the third question.

### 7.1 Model Exploration and Refitting-Based Methods

Refitting-based approaches assess feature importance by comparing predictive performance across alternative model specifications. In linear regression, partial or incremental $R^2$ measures the reduction in fit when a feature is removed and the model is refit, as discussed by Budescu (1993), Lindeman, Merenda, and Gold (1980), and Grömping (2007). Extensions such as dominance analysis average these effects across feature orderings.

These methods address a counterfactual question: how performance changes when a feature is removed and the model is re-estimated. As a result, they measure feature relevance or substitutability across models rather than contribution to realized fit within a fixed model.

Because refitting changes all model components, these measures are not additive and do not yield a decomposition of the predictive performance of a single fitted model. Computationally, they are expensive: evaluating feature importance requires repeated estimation across alternative feature sets, and in general scales poorly with the number of features.

## 7.2 Prediction Explanation Methods

Prediction-explanation methods attribute individual predictions to input features. Shapley-value approaches, introduced by Shapley (1953) and implemented for machine learning models by Lundberg and Lee (2017) as SHAP, define feature importance by averaging marginal contributions across subsets of features. Integrated gradients (IG), developed by Sundararajan, Taly, and Yan (2017), attribute predictions by integrating the gradient of the prediction function along a path from a baseline input to the realized input.

These methods explain predictions rather than predictive performance. A feature may have a large effect on predictions while contributing little, or even negatively, to predictive fit once redundancy and interactions are taken into account.

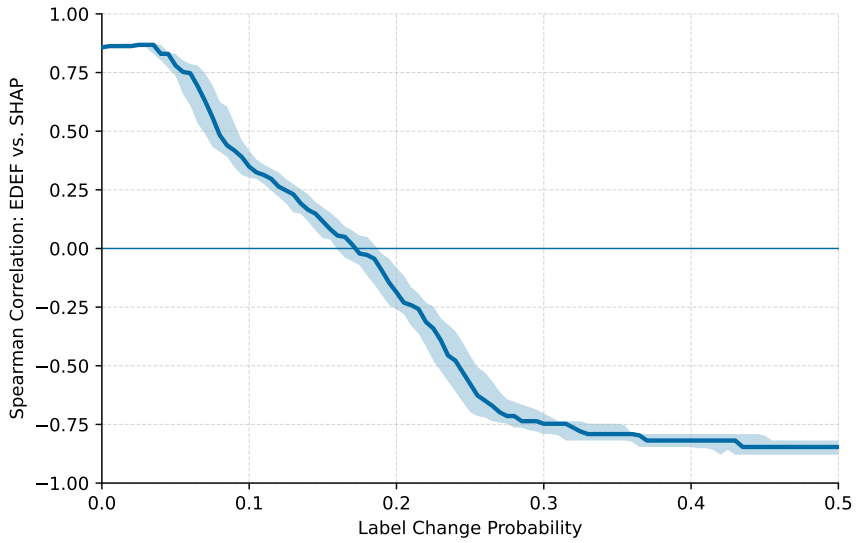
From a computational perspective, Shapley-based methods are costly. Exact computation requires $O(2^K)$ evaluations of the model and is infeasible except for very small $K$. Practical implementations rely on Monte Carlo sampling over feature orderings or coalitions, leading to costs of order $O(PK)$ for $P$ samples.

Gradient-based variants (e.g., DeepLIFT, DeepSHAP, or GradientExplainer in Lundberg and Lee (2017)) improve computational efficiency for differentiable models by replacing combinatorial perturbation with gradient-based approximations, but they attribute predictions rather than realized predictive accuracy.

Integrated gradients, developed by Sundararajan, Taly, and Yan (2017), attribute predictions by integrating the gradient of the prediction function along a path from a baseline input to the realized input. Specifically, they integrate $\nabla_x f(x(t))$ to decompose an individual prediction into feature-level contributions relative to the baseline.

From a computational perspective, integrated gradients are efficient. When automatic differentiation is available, each quadrature node requires one forward and one backward evaluation of the model, yielding total cost of order $O(M)$ per observation, where $M$ is the number of integration nodes. However, the resulting attribution reflects influence on predictions through $\nabla_x f(x)$ rather than contribution to predictive performance.

**Figure 1: Feature Importance Correlations**




The figure shows how the Spearman correlation between EDEF and SHAP feature importance for a classification model declines with lower model fit.

We fit a single L2 regularized logistic regression to a random half of the UCI adult income data. We then evaluate EDEF and SHAP feature importance on the hold-out half of the data, where we randomly switch outcome labels from 0 or 1 to the opposite value with probability $p$. This deliberately degrades the predictive accuracy of the fitted model for the hold-out sample.

Our empirical examples in section 8 compare several feature-importance measures. While the measures can align well, especially when predictive fit is high, the agreement declines systematically as fit deteriorates. Figure 1 illustrates this point for a classification model. The figure reports Spearman correlations between EDEF and SHAP feature importance evaluated in a hold-out sample for a fixed fitted model. At the far left, where predictive fit is strongest, the two measures have a correlation of 0.86. Moving to the right, we progressively corrupt the hold-out labels by independently switching binary outcomes from 0 to 1 or from 1 to 0 with increasing probability. This leaves the fitted model unchanged while degrading predictive fit in the evaluation sample. As predictive fit deteriorates, the agreement between the attribution methods declines sharply. In this example, the correlation falls to roughly half of its starting value at a switching probability of 0.09 and reaches zero at a switching probability of 0.17.

The agreement between prediction-based attribution methods such as SHAP and fit-based attribution methods such as EDEF tends to be high when predictive fit is very strong, but it can become weak, or even negative, when predictive fit is lower. For models with weaker initial predictive fit, the relation typically begins at lower correlation levels and declines more gradually.

## 7.3 Feature Reliance and Perturbation Methods

Perturbation-based methods assess feature importance by modifying inputs and observing changes in predictions or loss. Early implementations appear in Breiman (2001), and more recent analyses are provided by Fisher, Rudin, and Dominici (2019) and Gregorutti, Michel, and Saint-Pierre (2016). Common approaches include permutation importance and feature ablation.

These methods measure the sensitivity of model outputs or performance to input disruption and provide a notion of feature reliance. However, they do not yield an additive decomposition of predictive fit. The resulting importance measures depend on the perturbation scheme and generally do not sum to total performance. In the presence of correlated inputs, the interpretation becomes further dependent on the perturbation design.

Computationally, perturbation methods typically require $O(K)$ model evaluations per pass. To reduce noise, these evaluations are often repeated across multiple perturbations, leading to total cost $O(RK)$, with $R$ potentially large in high-dimensional settings.

## 7.4 Model-Fit Attribution

The framework developed here addresses the third question by attributing predictive fit directly to input features in a fixed, fitted model. Feature importance is defined as contribution to the reduction in expected loss relative to a baseline prediction.

Among existing approaches, SAGE (Covert et al., 2020) is closest in spirit, as it also provides a global, additive attribution of predictive performance. Hentschel (2026a) shows that SAGE and EDEF coincide under quadratic loss with an additive signal, but differ in general. SAGE evaluates model performance under counterfactual inputs generated by feature removal and measures how performance would change if features were unavailable. EDEF instead conditions on the realized inputs and attributes how observed feature variation contributed to realized predictive accuracy.

From a computational perspective, SAGE relies on Monte Carlo approximation of Shapley values and requires repeated evaluation of the model under counterfactual feature subsets. The resulting cost scales as $O(PK)$ for $P$ sampled coalitions. Each evaluation typically requires approximating conditional expectations of the model output given subsets of features, which introduces additional computational and statistical complexity. In practice, $P$ must be large to obtain stable estimates, making the method computationally intensive, particularly in high-dimensional settings.

Computationally, EDEF evaluates the gradient of the loss with respect to inputs along a path from baseline to realized inputs. Using a quadrature

rule with $M$ nodes, this requires $M$ forward and backward evaluations of the model when automatic differentiation is available, yielding total cost of order $O(M)$ per observation. Because the integration is one-dimensional in the path parameter, the method remains computationally tractable even for large $K$.

In the empirical examples in section 8, we show that EDEF is often orders of magnitude faster than SAGE. This is true even when we restrict SAGE to a small evaluation sample in order to make the evaluation time practical. By contrast, the EDEF examples always use the full evaluation sample. Although we can easily adapt EDEF to work on subsamples in order to speed calculations, this is generally not required except for extremely large evaluation samples.

The computational structure of EDEF is comparable to integrated gradients, which also require $M$ forward and backward passes per observation when automatic differentiation is available. The distinction is conceptual rather than computational: integrated gradients integrate $\nabla_x f(x(t))$ to explain predictions, whereas EDEF integrates $\nabla_x \ell(y, f(x(t)))$ to decompose predictive fit. Because the loss incorporates both the prediction and the realized outcome, the resulting attribution reflects contribution to predictive performance rather than influence on predictions alone. This distinction is consequential: a feature may strongly affect predictions through $\nabla_x f(x)$ while contributing little or negatively to predictive fit once weighted by the loss gradient.

Unlike refitting-based approaches, EDEF is model-conditional and does not require evaluating alternative specifications. Unlike prediction-explanation methods, it attributes predictive performance rather than individual predictions. Unlike perturbation methods, it yields an exact additive decomposition of explained fit.

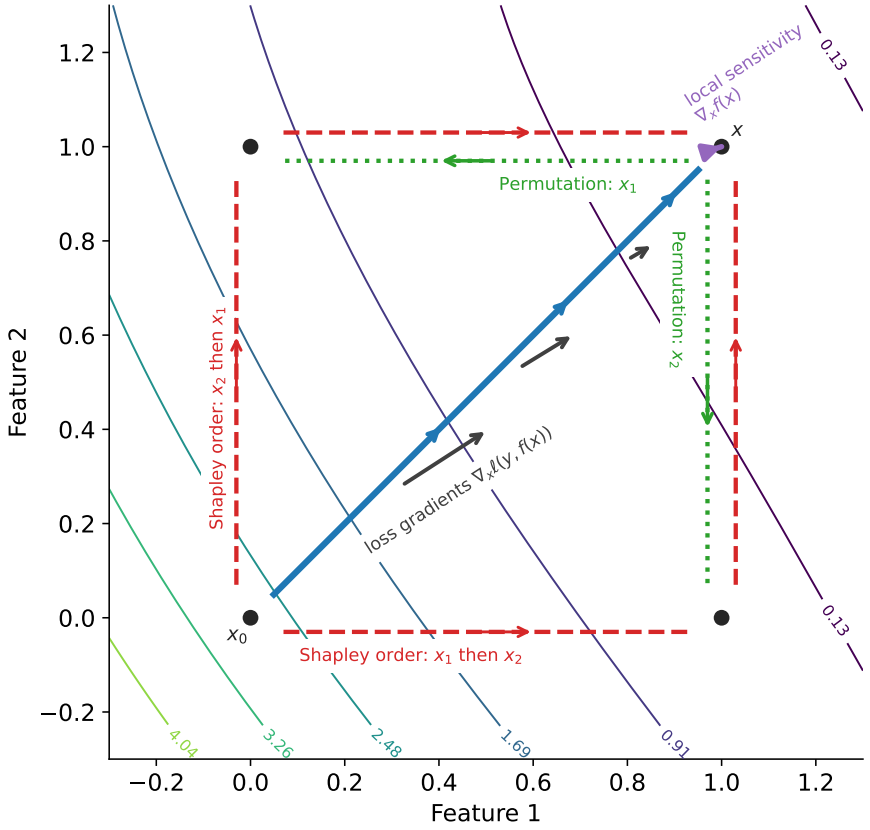
## 7.5 Geometric Perspective

Figure 2 illustrates these approaches in a bivariate nonlinear regression setting. Explained fit is defined as the reduction in loss when moving from the baseline input $x_0 = (0, 0)$ to the realized input $x = (1, 1)$. The background shows level curves of the loss surface, with lower values indicating better predictive fit.

The path-integral attribution evaluates the model along the straight-line path connecting $x_0$ to $x$, integrating loss gradients as the inputs move jointly from baseline to realized values. This captures how predictive performance evolves as features vary together and allocates interaction effects along the path.

By contrast, Shapley methods evaluate performance at corner points of the feature space, perturbation methods evaluate performance under modified

**Figure 2: Geometry of Feature Importance Measures**




The figure illustrates attribution of model fit in a bivariate nonlinear regression. The background shows level curves of the loss surface over the feature plane, with lower values indicating better fit. The solid blue diagonal line is the straight-line path used for the path-integral attribution, with loss gradients shown at selected nodes along the path. The dashed red paths depict Shapley attribution (without refitting), which evaluates the loss at corner points of the feature space across two feature orderings. The dotted green paths depict permutation attribution, which removes one feature at a time from $x = (1, 1)$. The purple arrow shows the local gradient-based sensitivity measure $\nabla_x f(x)$ evaluated at $x$.

inputs, and gradient-based methods focus on local behavior at a single point. These approaches rely on more limited information and do not directly capture how predictive fit is generated along continuous transitions in input space.

## 7.6 Summary Comparison

Across methods, there is a clear distinction between model exploration, prediction explanation, and model-fit attribution. These correspond to different objects and require different methodologies.

Refitting-based approaches provide insight into feature relevance across models but are computationally intensive and not additive. Prediction-

explanation methods are computationally efficient and widely applicable but focus on individual predictions rather than predictive performance. Perturbation methods measure feature reliance but depend on the perturbation design and do not yield additive decompositions.

The path-integral attribution developed here combines computational efficiency with an exact additive decomposition of predictive fit. By operating directly on the fitted model and integrating loss gradients along a path, it provides a principled and tractable solution to the problem of model-fit attribution in modern machine learning models.

# 8 Empirical Examples

This section illustrates the path-integral attribution in nonlinear regression and classification models. We use the same datasets as in the companion linear paper in order to isolate the effect of model class on predictive fit and on the resulting feature contributions.

The objective is not to optimize predictive performance, but to understand how different model classes generate predictive fit and how this is reflected in the attribution. This perspective follows the distinction between model evaluation and model tuning discussed by Hastie, Tibshirani, and Friedman (2009).

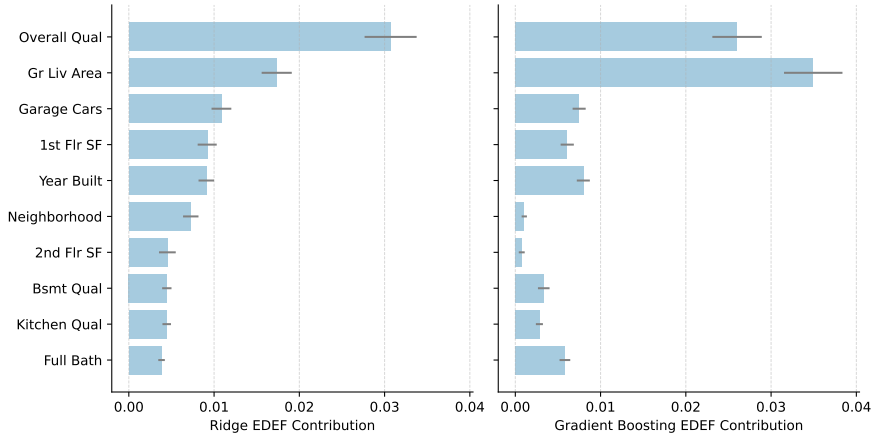
## 8.1 Regression Models

We evaluate several feature importance measures using the Ames housing dataset (De Cock, 2011), a common benchmark for tabular regression. We use the log sale price as the outcome. There are 2,930 total observations and 80 base features, including a mix of numerical variables and categorical attributes (e.g., neighborhood, quality ratings). We preprocess all variables using median imputation, standardization, and one-hot encoding. After one-hot encoding the categorical features, there are about 290 features.[4] When reporting feature importance, we group the contributions from all the dummies corresponding to a single categorical feature.

We estimate four regression models: a Ridge regression, a random forest model, a gradient boosting model, and a neural network with standard feedforward architecture. We train all models in a random half of the sample and then evaluate the models in the remaining half of the sample. For the Ridge regression, we choose regularization via cross validation. For the other models we use standard, default hyperparameters and confirm that

[4] The number of expanded features varies slightly across samples because we apply one-hot encoding after the train-test split, and rare categorical levels may be absent in a given training sample.

**Figure 3: EDEF: Ridge vs. Gradient Boosting**




The figure shows EDEF feature importance for the most important features in two regression models for the Ames housing data.

The left panel reports contributions to fit for the top features in a Ridge regression for the hold-out sample, with error bars indicating plus and minus two standard errors. The standard errors reflect sampling variability in the evaluation data, conditional on the fitted model, and therefore support statistical inference.

The right panel reports contributions to fit and standard errors for the same features in a gradient boosting model, for the same hold-out sample.

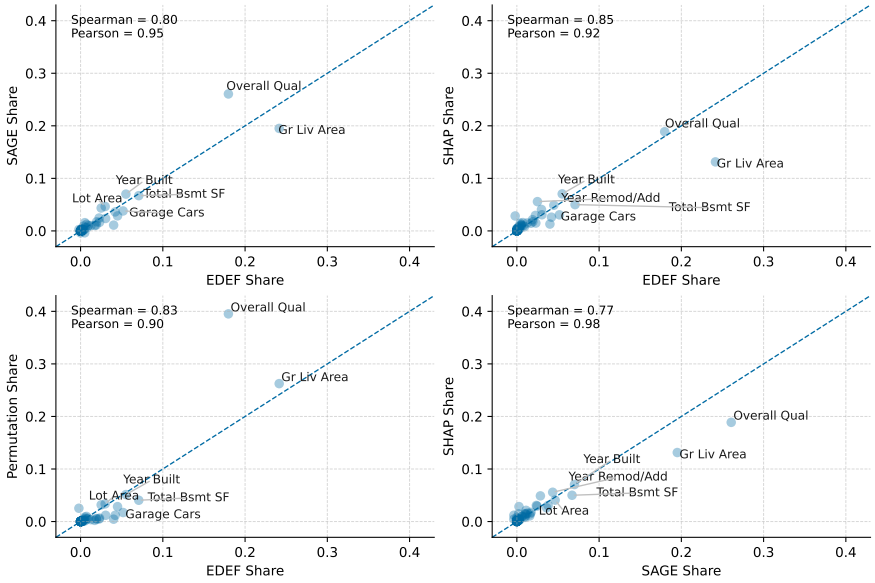
all models achieve similar in-sample fit. Across all models, we confirm that the EDEF attribution components sum to the total loss reduction relative to a baseline model.

For nonlinear models, we use the empirical mean of the evaluation sample inputs in the transformed feature space as the baseline. This choice provides a neutral anchor for the path decomposition and centers the attribution on deviations from the evaluation distribution. The baseline is not intended to represent a feasible observation or to reproduce the joint dependence structure of the inputs; it serves only as the reference point from which realized feature variation is measured.[5]

The left bar chart in figure 3 reports EDEF contributions in levels together with two-standard-error bands for the top 10 features based on the hold out sample for the Ridge regression. The top features make statistically significant contributions to model fit but the difference in importance of many adjacent features is statistically insignificant. The right bar chart reports feature importance and standard errors for the same features in the gradient

[5] In nonlinear models, the empirical feature mean need not map to the mean prediction because, generally, $f(E[X]) \neq E[f(X)]$. A more general baseline construction could instead define $x_0$ implicitly through a representative prediction condition such as $f(x_0) = E[y]$. We interepret the empirical mean baseline we use here as a simple and computationally convenient approximation to this broader class of representative-input constructions.

**Figure 4: Comparing Feature Importance: Gradient Boosting Regression**




The figure compares feature importance shares for a hedonic gradient boosting regression model using the Ames housing data. The EDEF and SAGE methods target predictive accuracy, with EDEF decomposing realized fit and SAGE evaluating counterfactual performance under feature removal. Permutation importance measures the change in performance under feature perturbations, while SHAP values reflect average contributions to predictions.

boosting model.[6]

This illustrates two important attributes of the EDEF framework. First, by interpreting contributions as sample averages we can construct standard errors and statistical inference in a straightforward manner. Second, by design EDEF feature importance is conditional on the fitted model and varies across models, even for the same outcome variable and feature set. The graph shows that there can be material differences across models.

Figure 4 reports pairwise comparisons of proportional contributions to model fit across methods for the gradient boosting model. The scatter plots show strong agreement between EDEF and SAGE, with points lying close to the 45-degree line. For linear regressions and quadratic loss EDEF coincides with the Shapley decomposition of model fit, and SAGE provides a Monte Carlo

[6] When feature contributions are evaluated for many components, testing for changes in importance across samples or models raises a multiple testing problem. In this situation, simple feature-by-feature inference can be misleading. Because EDEF contributions have standard errors that reflect sampling variability in the evaluation data, they can be combined with standard multiple-testing procedures, such as false discovery rate control (Benjamini and Hochberg, 1995; Benjamini and Yekutieli, 2001), to support valid inference across large numbers of features. Related issues arise in model selection and iterative refinement, where data-driven specification search can invalidate naive inference; these concerns are addressed by model-search procedures such as Hansen (2005).

approximation to the same object. Since the gradient boosting regression is nonlinear, this agreement is only approximate here. Although the two methods agree on the most important features, the Spearman rank correlation among the EDEF and SAGE feature importance measures is only 0.83 in this sample.

The other two methods, SHAP and permutations, address fundamentally different objects and we expect them to deviate from EDEF. Both methods tend to attribute higher importance to features that are correlated with other predictors, reflecting their reliance on perturbations or conditional expectations that break the correlation structure of the data. The apparent discrepancy reflects a difference in the question being answered. Overall Condition is indeed informative about house prices, but in this dataset it is strongly correlated with other structural features such as overall quality, age, and size. The fitted model distributes this information across multiple components. Once the full fitted signal is taken into account, Overall Condition contributes little additional alignment with the outcome. EDEF and SAGE attribute realized predictive accuracy to the components that actually generate it in the fitted model, and assign less credit to partly redundant features.

Table 1 shows how EDEF feature importance correlates with the other feature importance measures across different models. There is general agreement between EDEF and the other methods but that agreement is not always high. SAGE, which aligns perfectly with EDEF for linear regressions, materially deviates from EDEF for other, nonlinear models. The table shows Spearman rank correlations for the feature importance measures. Pearson correlations are qualitatively similar.

For neural-network regressions, SHAP produces highly compressed feature importance estimates, attributing similar importance to nearly all features. This results in weak agreement with EDEF, permutation importance, and SAGE, both in scikit-learn and in PyTorch gradient-based implementations. In contrast, EDEF, permutation importance, and SAGE remain broadly consistent with one another. This suggests that the discrepancy is not merely an implementation artifact of a particular SHAP explainer, but reflects a distinction between local prediction attribution and global decomposition of realized predictive performance.

## 8.2 Classification Models

We next consider a binary classification problem based on the UCI Adult income dataset (Dua and Graff, 2019; Kohavi, 1996). The dataset is a standard benchmark for binary classification. The outcome indicates whether annual

**Table 1: Correlation Among Feature Importance Measures**

| Model | Correlation | SAGE | SHAP | Permutation |
|---|---|---|---|---|
| **Panel A: Ames Housing Regression** | | | | |
| Ridge | Spearman | 0.97 | 0.61 | 0.72 |
| | Pearson | 1.00 | 0.78 | 0.93 |
| Random forest | Spearman | 0.95 | 0.96 | 0.96 |
| | Pearson | 0.96 | 0.94 | 0.94 |
| Gradient boosting | Spearman | 0.80 | 0.85 | 0.83 |
| | Pearson | 0.95 | 0.92 | 0.90 |
| Neural network | Spearman | 0.69 | 0.10 | 0.57 |
| | Pearson | 0.87 | -0.10 | 0.74 |
| Average | Spearman | 0.85 | 0.63 | 0.77 |
| | Pearson | 0.94 | 0.64 | 0.88 |
| **Panel B: UCI Adult Income Classification** | | | | |
| Logistic | Spearman | 0.93 | 0.86 | 0.91 |
| | Pearson | 0.95 | 0.93 | 0.81 |
| Random forest | Spearman | 0.84 | 0.79 | 0.81 |
| | Pearson | 0.69 | 0.71 | 0.75 |
| Gradient boosting | Spearman | 0.84 | 0.75 | 0.91 |
| | Pearson | 0.73 | 0.87 | 0.81 |
| Neural network | Spearman | 0.96 | 0.80 | 0.98 |
| | Pearson | 0.96 | 0.84 | 0.81 |
| Average | Spearman | 0.89 | 0.80 | 0.90 |
| | Pearson | 0.83 | 0.84 | 0.80 |

The table reports the correlations between the EDEF feature importance and SAGE, SHAP, and perturbation measures of feature importance, respectively.

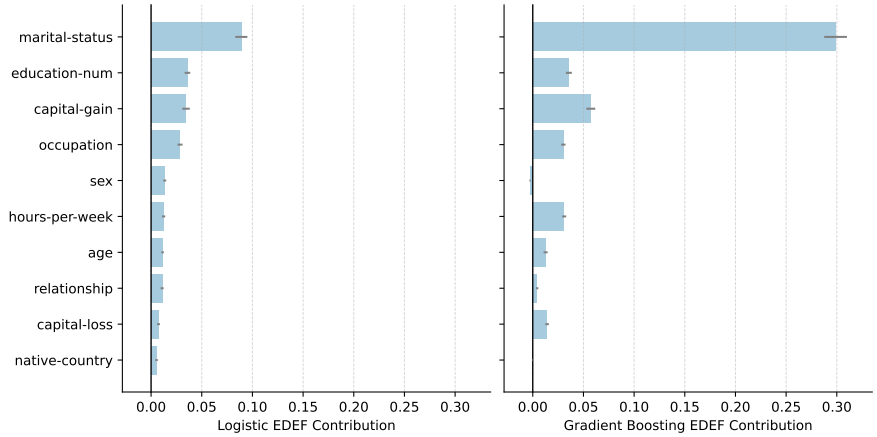
Panel A reports correlations for regression models applied to the Ames housing data. Panel B reports correlations for classification models applied to the UCI adult income data.

income exceeds $50,000. There are 48,842 total observations and 13 base features. Features include demographic and employment characteristics, preprocessed using median or mode imputation and one-hot encoding. After one-hot encoding the categorical features, there are about 100 features.

We estimate 4 classification models: An L2-regularized logistic regression, a random forest classifier, a gradient boosting classifier, and a neural network classifier using standard feedforward architecture. As for regression models, we train hyperparameters via cross-validation in a random half of the sample and then evaluate the models in the remaining half of the sample. For all models, we confirm that the EDEF attribution components sum to the total loss reduction relative to a baseline model. Once again, we use the feature means in the evaluation sample as the inputs for the baseline model.

Figure 5 compares the EDEF feature importance from the logistic regression

**Figure 5: EDEF: Logistic vs. Gradient Boosting**




The figure shows EDEF feature importance for the most important features in two classification models for the UCI adult income data.

The left panel reports contributions to fit for the top features in a L2-regularized logistic regression for the hold-out sample, with error bars indicating plus and minus two standard errors. The standard errors reflect sampling variability in the evaluation data, conditional on the fitted model, and therefore support statistical inference.

The right panel reports contributions to fit and standard errors for the same features in a gradient boosting classification model, for the same hold-out sample.

in the left panel to the feature importance from the gradient boosting model in the right panel. Since the two models fit different structures, even though they were trained on the same data, the model-conditional EDEF feature importance measures are not the same.

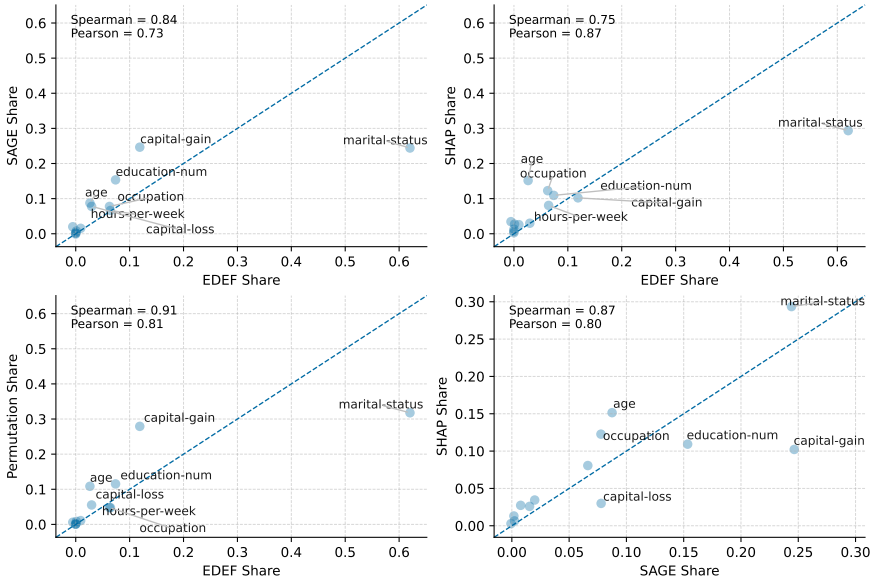
Figure 6 compares the proportional feature importance across EDEF, SAGE, SHAP, and perturbation models for the gradient boosting classifier. Here too, the feature importance measures have generally positive association, but they are not highly correlated with each other. EDEF and SAGE both investigate feature importance for model fit but come to different conclusions because we use neither quadratic loss nor a linear model. SHAP and perturbation methods answer different questions and naturally come to different conclusions.

## 8.3 Computational Comparison

Table 2 shows relative computation times for the feature importance measures in these applications.

We compute SHAP values using the `SHAP` Python package (Lundberg and Lee, 2017), SAGE values using the `SAGE-Importance` package (Covert et al., 2020), and permutation importance using the implementation in `Scikit-learn` (Breiman, 2001; Pedregosa et al., 2011). We generally use default settings. For SAGE, however, we use estimation samples of 250 observations for the Ames Ridge regressions and 1,000 observations for the

**Figure 6: Comparing Feature Importance for Classification**




The figure compares the feature importance shares for a logistic regression classification model using the UCI adult income data. The EDEF and SAGE methods target predictive accuracy, with EDEF decomposing realized fit and SAGE evaluating counterfactual performance under feature removal. Permutation importance measures the change in performance under feature perturbations, while SHAP values reflect average contributions to predictions.

UCI logistic regressions to speed up computations. All other methods use the full set of provided observations. We compute permutation importance by permuting raw input variables before preprocessing and use an average across 20 separate permutation samples.

The table shows that EDEF computes quickly. In these examples, EDEF is orders of magnitude faster than SAGE, which also decomposes model fit.

Although permutations and SHAP decompose predictions, not model fit, EDEF is also generally faster than either of these methods, in many cases much faster.

The EDEF run times for the tree-based models are materially reduced by the tree-specific integration approach we describe in section 5.2. The EDEF implementations are pure Python with localized Numba optimizations. Further speed gains may be possible in compiled implementations.

Table 2 also shows that the numerical line integral is greatly sped up by the automatic differentiation tools in PyTorch. For the neural network, applying direct quadrature requires roughly $O(MK)$ function calls, where $M$ is the number of quadrature points and $K$ is the number of expanded features. With automatic differentiation, we can reduce this to $O(M)$ function calls. This is reflected in the much faster EDEF calculations for the PyTorch neural

**Table 2: Runtime Comparison for Feature Importance Methods**

| | EDEF | SAGE | | SHAP | | Permutation | |
|---|---|---|---|---|---|---|---|
| Model | sec | rel | sec | rel | sec | rel | sec |
| **Panel A: Ames Housing Regression** | | | | | | | |
| Ridge | ( 0.0029) | 110,000 | ( 350) | 5.0 | ( 0.014) | 7,100 | (20) |
| RF | ( 0.35 ) | 7,200 | (2,500) | 62 | (22 ) | 240 | (82) |
| GBM | ( 0.026 ) | 74,000 | (1,900) | 9.8 | ( 0.25 ) | 1,600 | (42) |
| NN | (12 ) | 76 | ( 890) | 4.3 | (51 ) | 2.1 | (24) |
| NN-PT | ( 0.032 ) | | | 240 | ( 7.8 ) | | |
| **Panel B: UCI Adult Income Logistic Regression** | | | | | | | |
| Logistic | ( 0.029) | 30,000 | ( 850) | 6.8 | ( 0.19) | 380 | (11) |
| RF | ( 5.0 ) | 840 | (4,200) | 770 | (3,800 ) | 8.5 | (43) |
| GBM | ( 0.34 ) | 15,000 | (5,000) | 8.0 | ( 2.7 ) | 81 | (28) |
| NN | (66 ) | 40 | (2,700) | 5.1 | ( 340 ) | 0.23 | (15) |
| NN-PT | ( 0.27 ) | | | 830 | ( 230 ) | | |

The table reports relative timings of the feature importance calculations for the Ames housing regression models and the UCI adult income classification models. All values are rounded to two significant figures and are intended to convey approximate computational cost rather than precise measurements.

For EDEF, the table reports absolute compute time in seconds, in parentheses. For each alternative method, the table reports the multiple of the corresponding EDEF runtime for the same model and evaluation sample and the absolute time in parentheses.

The Ames housing Ridge regressions use 1,465 observations and 80 raw features. The UCI adult income logistic regressions use 24,421 observations and 13 base features. In each case, the number of observations represents a random hold out sample containing half of all available observations.

After one-hot encoding all categorical features, there are approximately 290 features in the Ames housing regressions and approximately 100 features in the UCI adult income logistic regressions. The number of expanded features varies slightly across samples because we apply one-hot encoding after the train-test split, and rare categorical levels may be absent in a given training sample.

To speed up computations, SAGE uses estimation samples of 250 observations for the Ames housing models and 1,000 observations for the UCI adult income models. All other methods use the full evaluation sample.

Permutation importance is computed by permuting raw input features before preprocessing. EDEF, SHAP, and SAGE are computed on the expanded post-preprocessing feature representation and then aggregated back to raw feature groups.

The last row in each panel reports selected timings for neural networks on PyTorch. The EDEF and SHAP times illustrate the speed gain available when we compute smooth neural-network gradients by automatic differentiation rather than finite differences.

network, shown in the last line of each panel. There are no PyTorch-specific implementations of SAGE and the permutation attribution. With the advantage of automatic differentiation, is materially faster than SAGE or permutations.

### 8.4 Discussion

The empirical results illustrate several general patterns. Nonlinear models can achieve similar predictive performance to linear baselines while redistributing feature contributions substantially, reflecting how different model classes allocate predictive fit across the input space. Features that strongly influence predictions do not necessarily contribute to predictive fit, and this divergence is more pronounced in models with strong interactions. Interaction effects play a central role in nonlinear attribution and are reflected in the path-dependent allocation of contributions across features.

These findings underscore the distinction between explaining predictions and explaining predictive performance, and highlight the value of model-conditional attribution for understanding how machine learning models generate fit.

## 9 Summary

This paper develops EDEF, a framework for attributing predictive fit to input features in fixed, fitted machine learning models. The central result is an exact, additive decomposition of explained fit based on a path integral of loss gradients in input space. In linear models, this reduces to closed-form Euler decompositions. In nonlinear models, the same logic requires integrating marginal contributions along a path from a baseline input to the realized input, preserving additivity while allocating interaction effects across features through an explicit path-based convention.

EDEF addresses a question that existing feature-importance methods do not: how much does each feature contribute to the realized predictive accuracy of a specific, fitted model? Prediction-explanation methods such as SHAP and integrated gradients are designed to attribute individual predictions, not predictive performance. Model-exploration methods such as partial $R^2$ and dominance analysis measure feature relevance across counterfactual model variants, not contribution to fit within a fixed model. Within the narrower and well-posed question of model-fit attribution, EDEF combines three properties that no existing method provides together: an exact decomposition with no residual terms, standard errors for each feature contribution derived from observation-level variation, and computational cost that is tractable across all major model classes. For tree-based models, the path integral reduces exactly to a weighted sum over split-crossing events. For differentiable models with automatic differentiation, the cost is $O(M)$ per observation regardless of the number of features.

The standard errors deserve particular emphasis. They are computed at essentially no additional cost once the path-integral contributions have been evaluated, and they are unavailable from any existing method for fit attribution. They allow feature contributions to be compared statistically across samples, model versions, and time periods, and they support formal inference about which features drive predictive performance. The empirical examples illustrate both uses: contributions with standard error bands in the cross-sectional analysis, and train-versus-holdout differences with combined standard errors that identify features whose contributions are stable out of sample.

The empirical results illustrate the practical consequences of the distinction between prediction explanation and fit attribution. Features that strongly influence predictions do not necessarily contribute to predictive fit, and this divergence grows with model nonlinearity. Even when two models achieve similar overall performance, their feature contributions under EDEF may differ materially, reflecting how different model classes allocate predictive fit across the input space.

Three properties of EDEF should be understood as deliberate design choices. First, the attribution is model-conditional: it describes how a specific fitted model generates fit, not whether a feature is intrinsically important. Different models fitted to the same data will produce different EDEF attributions, and this variation is informative rather than a defect. Second, the straight-line path treats all features symmetrically and provides a transparent convention for allocating interaction effects, but the allocation of those effects is path-dependent when interactions are strong. This dependence is explicit and shared by all path-integral methods, including integrated gradients. Third, the standard errors reflect sampling variability in the evaluation data, conditional on the fitted model; they do not incorporate uncertainty from model estimation. This is the appropriate inferential target when the model is treated as fixed and the evaluation sample is the source of variation.

Together with companion results for linear models in Hentschel (2026a), EDEF establishes a unified Euler-style perspective on feature importance across a broad class of predictive models. For understanding not just how well a model fits, but which features are responsible for that fit, EDEF provides an exact and computationally efficient solution, including standard errors.

## 10 References

Baydin, Atilim G., Barak A. Pearlmutter, Alexey A. Radul, and Jeffrey Mark Siskind, 2018, Automatic differentiation in machine learning: A survey, *Journal of Machine Learning Research* 18 (153), 1–43.

Belsley, David A., Edwin Kuh, and Roy E. Welsch, 1980, *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity* (Wiley, New York).

Benjamini, Yoav, and Yosef Hochberg, 1995, Controlling the false discovery rate: A practical and powerful approach to multiple testing, *Journal of the Royal Statistical Society, Series B* 57 (1), 289–300.

Benjamini, Yoav, and Daniel Yekutieli, 2001, The control of the false discovery rate in multiple testing under dependence, *Annals of Statistics* 29 (4), 1165–1188.

Breiman, Leo, 2001, Random forests, *Machine Learning* 45, 5–32.

Breiman, Leo, Jerome Friedman, Richard Olshen, and Charles Stone, 1984, *Classification and Regression Trees* (Wadsworth, Belmont, CA).

Budescu, David V., 1993, Dominance analysis: A new approach to the problem of relative importance of predictors in multiple regression, *Psychological Bulletin* 114 (3), 542–551.

Cortes, Corinna, and Vladimir Vapnik, 1995, Support-vector networks, *Machine Learning* 20 (3), 273–297.

Covert, Ian, Scott Lundberg, and Su-In Lee, 2021, Explaining by removing: A unified framework for model explanation, *Journal of Machine Learning Research* 22 (209), 1–90.

Covert, Ian C., Scott Lundberg, and Su-In Lee, 2020, Understanding global feature contributions with additive importance measures, in *NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems*, 17212–17223.

De Cock, Dean, 2011, Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project, *Journal of Statistics Education* 19 (3), 1–15.

Dua, Dheeru, and Casey Graff, 2019, UCI machine learning repository.

Fisher, Aaron, Cynthia Rudin, and Francesca Dominici, 2019, All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously, *Journal of Machine Learning Research* 20 (1), 1–81.

Friedman, Jerome H., 2001, Greedy function approximation: A gradient boosting machine, *The Annals of Statistics* 29 (5), 1189–1232.

Gneiting, Tilmann, and Adrian E. Raftery, 2007, Strictly proper scoring rules, prediction, and estimation, *Journal of the American Statistical Association* 102 (477), 359–378.

Gregorutti, Baptiste, Bertrand Michel, and Philippe Saint-Pierre, 2016, Correlation and variable importance in random forests, *Statistics and Computing* 27 (3), 659–678.

Griewank, Andreas, and Andrea Walther, 2008, *Evaluating Derivatives*, second edition (Society for Industrial and Applied Mathematics, Philadelphia, PA).

Grömping, Ulrike, 2007, Estimators of relative importance in linear regression based on variance decomposition, *The American Statistician* 61 (2), 139–147.

Hamilton, James D., 1994, *Time Series Analysis* (Princeton University Press, Princeton, NJ).

Hansen, Peter R., 2005, A test for superior predictive ability, *Journal of Business & Economic Statistics* 23 (4), 365–380.

Hastie, Trevor, and Robert Tibshirani, 1990, *Generalized Additive Models* (Chapman and Hall, New York, NY).

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman, 2009, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer Series in Statistics, second edition (Springer, New York).

Hentschel, Ludger, 2026a, Feature importance for predictive accuracy: An Euler decomposition, Working paper, Versor Investments, New York, NY.

Hentschel, Ludger, 2026b, TreeIG: Exact integrated gradients for tree-based models, Working paper, Versor Investments, New York, NY.

Kalman, Rudolf E., 1960, A new approach to linear filtering and prediction problems, *Transactions of the ASME–Journal of Basic Engineering* 82 (Series D), 35–45.

Kohavi, Ron, 1996, Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid, in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 202–207.

Lindeman, Richard H., Peter F. Merenda, and Ruth Z. Gold, 1980, *Introduction to Bivariate and Multivariate Analysis* (Scott, Foresman, Glenview, IL).

Lundberg, Scott M., and Su-In Lee, 2017, A unified approach to interpreting model predictions, in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 4768–4777 (Curran Associates Inc., Red Hook, NY).

Owen, Guillermo, 1977, Values of games with a priori unions, in Rudolf Henn, and Otto Moeschlin, eds., *Mathematical Economics and Game Theory*, volume 141 of *Lecture Notes in Economics and Mathematical Systems*, 76–88 (Springer, Berlin, Germany).

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, 2011, Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* 12, 2825–2830.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams, 1986, Learning representations by back-propagating errors, *Nature* 323, 533–536.

Schölkopf, Bernhard, and Alexander J. Smola, 2002, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* (MIT Press, Cambridge, MA).

Shapley, Lloyd S., 1953, A value for $n$-person games, *Contributions to the Theory of Games* 2, 307–317.

Silberberg, Eugene, 1978, *The Structure of Economics: A Mathematical Analysis* (McGraw–Hill, New York, NY).

Sundararajan, Mukund, Ankur Taly, and Qiqi Yan, 2017, Axiomatic attribution for deep networks, in Doina Precup, and Yee Whye Teh, eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 3319–3328 (PMLR).

Tasche, Dirk, 2008, Capital allocation to business units and sub-portfolios: The Euler principle, in Andrea Resti, ed., *Pillar II in the New Basel Accord: The Challenge of Economic Capital*, 423–453 (Risk Books, London, England).

Vapnik, Vladimir N., 1998, *Statistical Learning Theory* (Wiley, New York, NY).

Wahba, Grace, 1990, *Spline Models for Observational Data* (Society for Industrial and Applied Mathematics, Philadelphia, PA).

Wooldridge, Jeffrey M., 2002, *Econometric Analysis of Cross-Section and Panel Data* (MIT Press, Cambridge, MA).

# A Standard Errors

This appendix derives standard errors for the Euler contributions to explained fit in the nonlinear regression setting. The derivation parallels the linear and classification cases and requires no additional assumptions beyond those already imposed in the main text.

## A.1 Observation-Level Contributions

Recall that explained fit is defined as the reduction in average loss relative to the baseline prediction,

$$\Delta\mathcal{L} = \mathcal{L}(\overline{y}) - \mathcal{L}(\widehat{y}(X)), \tag{47}$$

and that the path-integral construction yields the additive decomposition

$$\Delta\mathcal{L} = \sum_{j=1}^{K} C_j, \tag{48}$$

with component-level contributions

$$C_j = -\mathbb{E}\left[(x_j - x_{0j}) \int_0^1 \frac{\partial}{\partial x_j} \ell(y, f(x(t)))\, dt\right]. \tag{49}$$

Define the corresponding observation-level contribution for observation $i$ as

$$c_{ij} = -(x_{ij} - x_{0j}) \int_0^1 \frac{\partial}{\partial x_j} \ell\big(y_i, f(x_i(t))\big)\, dt. \tag{50}$$

By construction,

$$C_j = \mathbb{E}[c_{ij}] = \frac{1}{N} \sum_{i=1}^{N} c_{ij}. \tag{51}$$

In practice, the integral over $t$ is evaluated numerically using a fixed quadrature rule. This numerical approximation does not affect the asymptotic logic of the standard errors.

## A.2 Standard Errors

Treating the fitted model parameters as fixed, the Euler contribution $C_j$ is a sample average of the observation-level quantities $c_{ij}$. Under standard

regularity conditions,

$$\sqrt{N}\,(C_j - \mathbb{E}[c_{ij}]) \xrightarrow{d} \mathcal{N}\big(0,\, \mathbb{E}\big[(c_{ij} - C_j)^2\big]\big)\,. \tag{52}$$

Accordingly, the standard error of $C_j$ is

$$SE(C_j) = \sqrt{\frac{1}{N}\,\mathbb{E}\big[(c_{ij} - C_j)^2\big]}. \tag{53}$$

In empirical applications, this quantity is estimated by

$$\widehat{SE}(C_j) = \sqrt{\frac{1}{N(N-1)}\sum_{i=1}^{N}(c_{ij} - C_j)^2}. \tag{54}$$

This estimator is identical in form to the standard errors used in the linear and classification settings. No modification is required for the nonlinear case.

## A.3 Standard Errors and Grouped Contributions

Let $c_i = (c_{i1}, \ldots, c_{iK})^\top$ denote the vector of observation-level contributions, so that the global contribution of component $j$ is

$$C_j = \mathbb{E}[c_{ij}], \tag{55}$$

with expectations understood as sample averages. Throughout, the fitted model is treated as fixed, and randomness arises only from sampling across observations.

Because $C_j$ is an average of observation-level contributions, its standard error can be computed directly from the sample variance of $c_{ij}$,

$$SE(C_j) = \sqrt{\frac{1}{N}\,\mathbb{E}\big[(c_{ij} - C_j)^2\big]} = \sqrt{\frac{1}{N(N-1)}\sum_{i=1}^{N}(c_{ij} - C_j)^2}. \tag{56}$$

This expression does not require estimation or storage of any covariance matrix and remains numerically stable even when the number of components is large.

For any group of components $G \subset \{1, \ldots, K\}$, define the observation-level group contribution

$$c_{iG} = \sum_{j \in G} c_{ij}, \tag{57}$$

and the corresponding global group contribution

$$C_G = \mathbb{E}[c_{iG}] = \sum_{j \in G} C_j. \tag{58}$$

The standard error of the grouped contribution is obtained directly from the sample variance of $c_{iG}$,

$$SE(C_G) = \sqrt{\frac{1}{N}\,\mathbb{E}[(c_{iG} - C_G)^2]} = \sqrt{\frac{1}{N(N-1)} \sum_{i=1}^{N} (c_{iG} - C_G)^2}. \tag{59}$$

This univariate computation automatically accounts for correlation among components within the group and avoids forming any high-dimensional covariance objects.

Alternatively, we can define the covariance matrix of the vector of global contributions $C$ as

$$\widehat{\Sigma}_C = \frac{1}{N(N-1)} \sum_{i=1}^{N} (c_i - C)(c_i - C)^\top. \tag{60}$$

Then for any group $G$, the grouped standard error can equivalently be written as

$$SE(C_G) = \sqrt{\mathbf{1}_G^\top \widehat{\Sigma}_C \mathbf{1}_G}, \tag{61}$$

where $\mathbf{1}_G$ is the indicator vector for the group. This expression is algebraically identical to the univariate variance formula above. In practice, however, computing $SE(C_G)$ from the observation-level group contributions $c_{iG}$ is simpler, faster, and more robust, especially when the number of components is large.

## A.4 Interpretation

These standard errors quantify sampling variability in the Euler attribution conditional on the fitted prediction function. As a result, they do not incorporate uncertainty in model estimation.

The observation-level formulation makes it straightforward to compute standard errors, confidence intervals, and group-level inference for nonlinear Euler attributions at essentially no additional computational cost once the path integrals have been evaluated.

# B Decomposition Algorithm

This appendix outlines the algorithm for computing exact Euler-style contributions to explained fit in nonlinear machine-learning regression or classification models.

The pseudo-code uses matrix notation for clarity and computational efficiency. Rows of $X \in \mathbb{R}^{N \times K}$ correspond to realized input vectors $x_i$, while the baseline input $x_0$ is a single $K$-vector replicated across observations. The output $C_j$ corresponds to the contribution defined in the main text and reflects the average contribution to model fit across the full sample $y$.

We can compute gradients with respect to inputs via finite differences but automatic differentiation is more efficient. Under a weighted or GLS loss, we evaluate all quantities in the transformed space but attribution always is with respect to the original input coordinates.

**Algorithm 1: Path-integral attribution of explained fit for a scalar loss**

```
# Notation mapping to main text:
# - X[i,:]     corresponds to x_i (realized inputs for observation i)
# - x0         corresponds to x_0 (baseline input in feature space)
# - y_hat[i]   corresponds to \widehat y(x_i)
# - C[j]       corresponds to C_j
# - c[i,j]     corresponds to c_{ij} (observation-level contribution)
# - dL         corresponds to \Delta \mathcal L

# Inputs:
# y       : (N,) outcomes or labels
# X       : (N, K) inputs
# f       : prediction function; y_hat = f(X)
# loss    : scalar loss function; ell(y_i, y_hat_i)
# grad_x_loss :
#          routine returning row-wise gradient of loss w.r.t. inputs
#          G[i,j] = d ell(y_i, f(x_i)) / d x_{ij}, shape (N,K)
#          (via autodiff or chain rule)
# x0      : (K,) baseline input (default: zeros(K))
# M       : number of quadrature nodes on [0,1]
# groups  : optional list of feature index sets

# Outputs:
# C       : (K,) global feature contributions
# SE      : (K,) standard errors
# dL      : scalar explained loss improvement
# (optional)
# C_G     : grouped contributions
# SE_G    : grouped standard errors

if x0 is None:
```

```
    x0 = zeros(K)

# Baseline inputs and prediction
X0    = repeat_row(x0, N)     # (N,K)
y0_hat = f(X0)                # (N,)

# Loss at baseline
L0 = mean(loss(y, y0_hat))

# Loss at realized inputs
y_hat = f(X)
L     = mean(loss(y, y_hat))

# Explained fit
dL = L0 - L

# Quadrature nodes and weights on [0,1]
t_grid, a_grid = quadrature_nodes_weights_on_0_1(M)

# Observation-level contributions
c = zeros((N, K))

for m in range(M):
    t = t_grid[m]
    a = a_grid[m]

    Xt = X0 + t * (X - X0)    # (N,K)

    y_hat_t = f(Xt)           # (N,)

    # Gradient of loss w.r.t. inputs
    G = grad_x_loss(y, Xt, y_hat_t) # (N,K)

    # Accumulate contributions
    # Minus sign reflects dL = L0 - L
    c += -a * (X - X0) * G

# Global contributions
C = mean_over_i(c)            # (K,)

# Sanity check:
# sum(C) ~= dL

# Standard errors
dc = c - C[None, :]
var_c = sum_over_i(dc ** 2) / (N - 1)
SE = sqrt(var_c / N)

# Optional grouped contributions
```

```
if groups is not None:
    G = len(groups)
    C_G = zeros(G)
    SE_G = zeros(G)
    for g in range(G):
        idx = groups[g]

        c_g = sum_over_j(c[:, idx])
        C_G[g] = mean(c_g)

        dc_g = c_g - C_G[g]
        var_g = sum(dc_g ** 2) / (N - 1)
        SE_G[g] = sqrt(var_g / N)

# Notes:
# - The algorithm applies to any differentiable scalar loss.
# - grad_x_loss can be obtained via automatic differentiation or chain rule:
#     grad_x_loss = (d ell / d y_hat) * (d f / d x)
# - For squared error: d ell / d y_hat = -2 (y - y_hat)
# - For log loss:   d ell / d score = (p - y)
# - Standard errors treat the fitted model as fixed.
```