

Model Fingerprint: General and Efficient Interpretability from Fundamental Prediction Logic

THIS VERSION: January 28, 2026

Yimou Li

Vice President

State Street Associates

yli24@statestreet.com

Yin Li

Assistant Vice President

State Street Associates

yli31@statestreet.com

David Turkington

Senior Managing Director

State Street Associates

dturkington@statestreet.com

Abstract

We show that the model fingerprint framework of Li, Turkington, and Yazdani (2020) matches the aggregate feature importance scores of Lundberg and Lee's (2017) SHAP measure, but the model fingerprint method's focus on logical components further enables an effective approximation approach with polynomial (as opposed to exponential) computation complexity. Overall, the model fingerprint approach provides several advantages: (1) it delivers explicit attribution of linear, nonlinear, pairwise interaction and higher-order interaction effects, (2) it allows for much faster computation of lower-order effects, and (3) it offers a direct and intuitive explanation of prediction logic.

Model Fingerprint: General and Efficient Interpretability from Fundamental Prediction Logic

Most practitioners rightfully demand some level of human interpretability from complex prediction routines. Interpretability is particularly important to avoid the pitfalls of underfitting, overfitting, or otherwise bad-fitting of models for applications where the quality of the prediction output is difficult or impossible to judge before the unknown outcome occurs. One approach to interpretability is to measure the overall importance of a given input feature to a prediction. The Shapley value, introduced by Shapley (1951) for cooperative game theory, provides an appealing theoretical foundation for feature value attribution because it is the unique measure that satisfies certain fair attribution properties. Its computation requires defining a value function for the outcome of a game when the game is played by every combinatorial subset of players.

Lundberg and Lee (2017) proposed the SHAP (SHapley Additive exPlanation) method to apply the Shapley value in the context of machine learning. SHAP considers input features as players, and, to avoid retraining variants of a model based on every combinatorial subset of the model's input variables, SHAP approximates how a model would predict based on a subset of features using a conditional expectation of the full prediction function as its value function. The computation of the SHAP attribute is still expensive as it requires evaluating 2^n subset models for n features.

Li, Turkington, and Yazdani (2020) proposed an alternative interpretability framework called model fingerprint which reveals a visual characteristic pattern of logical component contributions for a prediction function. The model fingerprint method measures the incremental effect of first order, second order, and higher order effects on predictions. Like SHAP, it uses a conditional expectation of the full prediction function. By varying some input feature values systematically, model fingerprint quantifies how features and their interactions affect prediction values. For example, we can vary the input values for individual features first. Then, we can vary the joint occurrence of pairs of features and measure their interaction effect above and beyond their combined independent effects. We can proceed to measure successive higher-order interactions in excess of all the lower-order effects. The resulting map of incremental effects can be used to trace the conditional logic of any individual prediction, to summarize the aggregate importance of logical components across many prediction tasks, and to summarize the average importance of individual features for specific predictions or across many tasks.

In this paper, we prove that the aggregate feature importance scores generated by model fingerprint for individual prediction tasks are equivalent to those generated by SHAP. Therefore, model fingerprint enjoys the same desirable properties as SHAP. Because SHAP measures a feature’s contribution by repeatedly omitting it from subgroups, it does not lend itself to decomposing the nature of a feature’s contribution, and its calculation routine is lengthy and indirect. Thus, model fingerprint provides additional benefits that are not inherent to SHAP:

1. Model fingerprint explicitly reveals the importance of individual feature effects, pairwise interaction effects, and higher-order interaction effects. Individual feature effects can be further decomposed into linear and nonlinear components. For interpretation, it is helpful to distinguish between features that have a direct linear impact on predictions versus those that contribute mainly by conditional other relationships. When asked to explain their reasoning, people are accustomed to describing the nature of the relationship they presume for a feature, not just its total importance. Conditional relationships are core to model logic and narrative-style explanations. If we know which feature co-occurrences matter, we can overlay prior beliefs on a model to gauge our trust in its learned prediction rules. And we can distinguish between two instances when a feature mattered, but for different reasons.
2. Model fingerprint can efficiently compute lower-order effects for individual features and pairwise interactions to reveal important components of prediction logic quickly. Although higher-order combinatorics require exponentially increasing computation time, higher-order effects are of diminishing importance for many models.
3. Model fingerprint is more straightforward to explain and understand. The core principle is the same as understanding coefficients in a linear regression analysis, where we “hold all else equal” and transparently measure one logical component effect at a time.

The remainder of the paper is structured as follows. First, we describe the model fingerprint method in detail. Next, we describe the Shapley value and SHAP. We then prove the equivalence of summary feature importance scores for model fingerprint and SHAP, and we illustrate conceptual and computational benefits of model fingerprint including a low-order approximation method. Lastly, we conclude.

Setup and Notation

Let $N = \{1, \dots, n\}$ index the features of X in a prediction function $\hat{f}(X)$ that yields a scalar prediction for any input vector. For a specific input x , let x_S represent a subset of the input values for a

collection of features $S \subseteq N$ containing $|S|$ items. Denote by $S \cup \{a\}$ the set S explicitly augmented with feature a and $S \setminus \{a\}$ the set S explicitly omitting feature a .

Definition 1 (Centered partial prediction function). The centered conditional expectation partial prediction function is:

$$\hat{f}_S(x_S) \equiv \mathbb{E}[\hat{f}(X) \mid X_S = x_S] - \mathbb{E}[\hat{f}(x)] \quad (1)$$

Equation 1 renders a prediction based only on specific inputs for the features indexed by S in a way that is permitted to condition on the values of x_S . The first expectation aggregates over values of the unspecified complementary features $N \setminus S$ and the second expectation is equal to taking the expected value over S of the first term to center it, $\mathbb{E}_S [\mathbb{E}_{N \setminus S}[\hat{f}(X) \mid X_S = x_S]] = \mathbb{E}[\hat{f}(x)]$. Note that we leave ambiguous the specific method for estimating $\hat{f}_S(x_S)$ from an empirical data sample, as this process admits multiple possibilities, which we will discuss later.

Lastly, for notational concision, we will often omit the argument of the partial prediction function because it always matches the superscript of the function itself, thus $\hat{f}_S = \hat{f}_S(x_S)$. And based on the definition, we have $\hat{f}_\emptyset = 0$.

Model Fingerprint Methodology

Originally introduced by Li, Turkington, and Yazdani (2020), model fingerprint is a model-agnostic interpretability framework that visually summarizes a prediction function's relative reliance on various logical components of its inputs. It quantifies the incremental effects of first-order, second-order, and higher-order interactions in the same units as the fitted model's overall predictions.

For a prediction input x , the first-order effect for feature a is given by the centered partial prediction. The second-order effect for a pair of features a and b is given by the pair's centered partial prediction minus the sum of their independent effects. The third-order effect for a set of features a , b , and c , is measured incremental to all of the independent and pairwise effects of those features, and so on.

$$\psi_a = \hat{f}_a \quad (2)$$

$$\psi_{a,b} = \hat{f}_{a,b} - (\psi_a + \psi_b) \quad (3)$$

$$\psi_{a,b,c} = \hat{f}_{a,b,c} - (\psi_{a,b} + \psi_{a,c} + \psi_{b,c}) - (\psi_a + \psi_b + \psi_c) \quad (4)$$

We now state this definition generally.

Definition 2 (Model fingerprint logical interaction components). The incremental interaction components ψ_T for any nonempty $T \subseteq N$ is the residual of \hat{f}_T after subtracting all nonempty lower-order components:

$$\psi_T \equiv \begin{cases} \hat{f}_T - \sum_{U \subset T} \psi_U & T \neq \emptyset \\ 0 & T = \emptyset \end{cases} \quad (5)$$

Equivalently, we may write ψ_T explicitly in terms of positive and negative \hat{f} terms, where $t = |T|$ and $u = |U|$ represent the size of the respective set collections:

$$\psi_T = \begin{cases} \sum_{U \subseteq T} (-1)^{t-u} \hat{f}_U & T \neq \emptyset \\ 0 & T = \emptyset \end{cases} \quad (6)$$

Because the model fingerprint terms are computed bottom-up starting with the lowest order terms, it is not strictly necessary to calculate every high order effect explicitly. If desired, we can simply compute the leftover value of all higher order effects as the remaining residual of \hat{f} . We will make use of this fact for computational expediency later.

The bottom-up accounting mechanism of model fingerprint facilitates the following convenient explanatory elements. First, we may summarize the importance of each effect for a broad range of prediction inputs by computing the expected absolute value of each component.

$$\text{Overall independent effect of } a = \mathbb{E}_a[|\psi_a|] \quad (7)$$

$$\text{Overall pairwise effect of } a, b = \mathbb{E}_{a,b}[|\psi_{a,b}|] \quad (8)$$

$$\text{Overall high order effects of } a, b, c = \mathbb{E}_{a,b,c}[|\psi_{a,b,c}|] \quad (9)$$

Second, if desired, we may further decompose the independent effects of features into linear and nonlinear components. We do so by fitting the linear model $\hat{f}_a(X_a) = \alpha + \beta X_a + \epsilon$ with ordinary least squares to estimate β and project the linear component of each ψ_a as $\hat{\iota}_a$. Now we can define overall linear and nonlinear independent effects.

$$\text{Overall linear independent effect of } a = \mathbb{E}_a[|\hat{\iota}_a|] \quad (10)$$

$$\text{Overall nonlinear independent effect of } a = \mathbb{E}_a[|\psi_a - \hat{\iota}_a|] \quad (11)$$

Rather than merely ranking feature importance, this approach provides a nuanced picture of how features influence predictions—capturing linear effects, nonlinear relationships, and pairwise interactions that reveal the underlying patterns driving the model’s decisions. Intuitively,

when a model has learned a purely linear relationship between a feature and its prediction target, the nonlinear and interaction effects will be zero.¹

Exhibits 1, 2, and 3 provide a visual illustration of the model fingerprint for a contrived example. In Exhibit 1, for a selected feature, the conditional expectation (dark blue line) is plotted alongside its linear fit (shown as a light blue dotted line). The linear effect is quantified by the average absolute deviation from the linear fit, while the nonlinear effect is captured by the average absolute deviation of the conditional expectation from the linear fit. The shaded region in the rightmost panel underscores the magnitude of the nonlinear component. Exhibit 2 shows the contributions of the linear and nonlinear components for one prediction instance. Exhibit 3 shows pairwise interaction effects in excess of independent effects for two features.

Exhibit 1: Linear and Nonlinear Effect of an Individual Feature

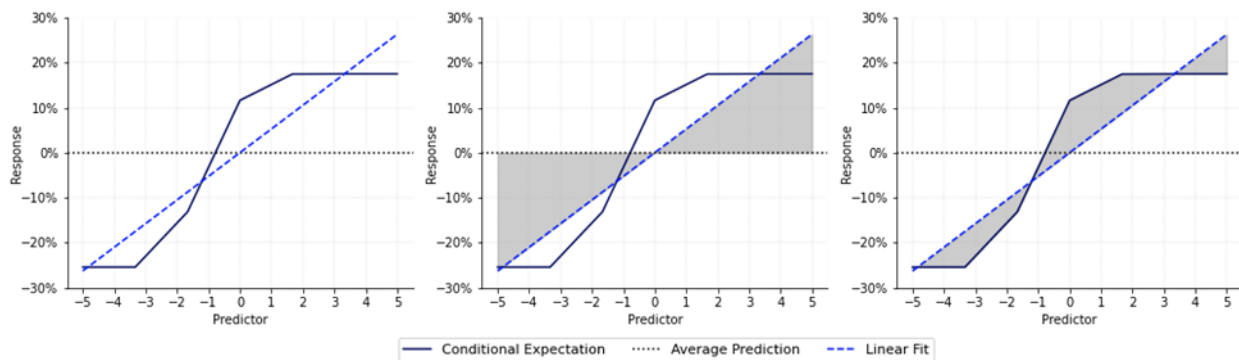
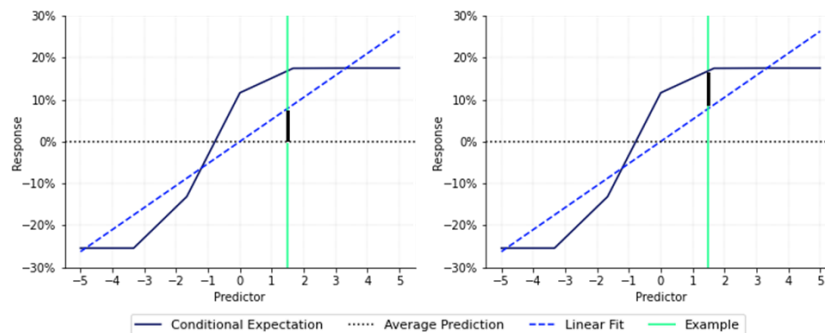
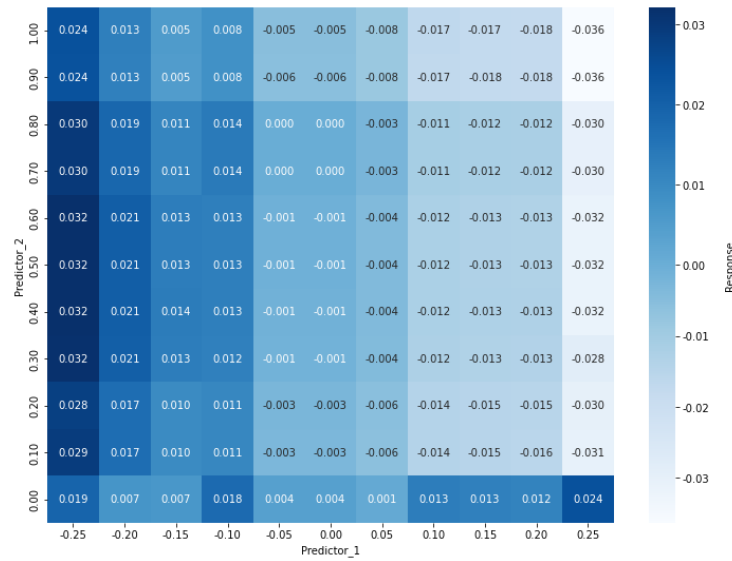


Exhibit 2: Linear and Nonlinear Effect of a Specific Input Value of an Individual Feature



¹ There are, however, other scenarios in which interaction effects will be measured as nonzero even for functions that can be expressed solely in terms of independent effects. We will discuss these scenarios in a later section.

Exhibit 3: Interaction Effect Between Two Features



SHAP Methodology

We aim to compare and contrast the model fingerprint method with the SHAP method. Originally introduced by Lundberg and Lee (2017), SHAP attributes importance to each input feature in a prediction function by applying the Shapley (1951) attribution framework from cooperative game theory. SHAP summarizes importance at the level of each feature by evaluating the impact of adding the target variable to every other possible set of features. We first describe the classical Shapley value, then its implementation via SHAP.

Shapley Value

The Shapley (1951) value is a concept in cooperative game theory. It allocates the total gains of a coalition of players fairly among the members, where fairness is specified as adherence to four principled axioms:

- **Efficiency:** The total payoff is distributed completely among the players.
- **Symmetry:** Players who contribute equally receive equal payments.
- **Dummy:** A player who adds no value to any coalition receives a value of zero.
- **Additivity:** The Shapley value for the union of two games is the sum of the Shapley values for each game.

The Shapley value is the unique allocation guaranteed to satisfy all four principles for any game. For a given player a in a game with n players, it is computed as an aggregation of the marginal effects of adding player a to every possible starting coalition of players, where the value function $v(\cdot)$ represents the game's total payoff for a given set of players.

$$\phi_a^{Shapley} = \sum_{S \subseteq N \setminus \{a\}} \frac{|S|!(n-|S|-1)!}{n!} (v(S \cup \{a\}) - v(S)) \quad (12)$$

To demonstrate how the Shapley value works in practice, consider a prediction function for house prices with three input variables: location (L), floor plan size (F), and age of the house (A). Let $v(\cdot)$ denote the predicted house price given a specific set of input variable values. Suppose we are interested in a prediction for L = "Downtown", F = 1000 sqft, and A = 5 years. Exhibit 4 presents the predicted prices for all possible coalitions of input variables, along with the marginal contribution of input variable L = "Downtown" in each case.

Exhibit 4: Predicted Price and Marginal Contributions of L = "Downtown" for Shapley Value Calculation

Feature Set	Prediction (\$, '000)	Increment from adding L (\$, '000)
{ }	250	-
{ L = "Downtown" }	300	50
{ F = 1000 sqft }	270	-
{ A = 5 yrs }	260	-
{ L = "Downtown", F = 1000 sqft }	350	80
{ L = "Downtown", A = 5 yrs }	320	60
{ F = 1000 sqft, A = 5 yrs }	290	-
{ L = "Downtown", F = 1000 sqft, A = 5 yrs }	380	90

The marginal increase in the third column reflects how much the predicted price increases when L = "Downtown" is added to an existing coalition. Based on these values, the Shapley value for L = "Downtown" is calculated by averaging its marginal contributions across all coalitions:

$$\phi_L^{Shapley} = \frac{0!(3-0-1)!}{3!} \times 50 + \frac{1!(3-1-1)!}{3!} \times 80 + \frac{1!(3-1-1)!}{3!} \times 60 + \frac{2!(3-2-1)!}{3!} \times 90 = 70$$

We can perform similar calculations for F = 1000 sqft and A = 5 years as well, yielding $\phi_F = 40$ and $\phi_A = 20$. Additionally, the baseline of Shapley value $\phi_\emptyset^{Shapley} = 250$ which is the average of all prediction. Then the total predicted price can thereby be decomposed as:

$$\phi_A^{Shapley} + \phi_F^{Shapley} + \phi_L^{Shapley} + \phi_\emptyset^{Shapley} = 20 + 40 + 70 + 250 = 380$$

Despite the Shapley value's theoretical virtues, it is important to note that there is no unique choice of value function for the purpose of interpreting a prediction function. The most theoretically obvious choice is to define the value function as the prediction made by an alternate prediction function (model) trained on the subset of features. While conceptually clean, this approach is extremely computationally prohibitive because it requires retraining an entire model for an exponentially large (2^n) number of feature combinations. It is almost never feasible in practice.

SHAP

The SHAP method, which is shorthand for SHapley Additive exPlanations, applies the Shapley formula with the value function equal to the partial prediction function described earlier. This choice avoids the prohibitive need to retrain 2^n distinct models for n features, though it must still evaluate 2^n distinct comparisons.

$$\phi_a^{SHAP} = \begin{cases} \sum_{S \subseteq N \setminus \{a\}} \frac{|S|!(n-|S|-1)!}{n!} (\hat{f}_{S \cup \{a\}} - \hat{f}_S), & a \in N \\ \mathbb{E}[\hat{f}(x)], & a = \emptyset \end{cases} \quad (13)$$

In the same spirit as equation 6, we may write the SHAP value equivalently as an explicit sum of \hat{f}_U terms. Let $u = |U|$.

$$\phi_a^{SHAP} = \begin{cases} \sum_{U \subseteq N} c_a^{SHAP}(U) \hat{f}_U, & a \in N \\ \mathbb{E}[\hat{f}(x)], & a = \emptyset \end{cases} \quad (14)$$

$$c_a^{SHAP}(U) = \begin{cases} \frac{(u-1)!(n-u)!}{n!}, & a \in U \\ -\frac{u!(n-u-1)!}{n!}, & a \in N \setminus U \end{cases} \quad (15)$$

Equations 14 and 15 follow from considering two separate scenarios for U . If $a \notin U$, we set $U = S$ in equation 13 and determine that there is a single term of \hat{f}_U with a negative coefficient. If $a \in U$, we set $U = S \cup \{a\}$ which means that $S = U \setminus \{a\}$, and determine that the term \hat{f}_U appears with a coefficient in which S is decremented by one unit, hence $\frac{(u-1)!(n-u)!}{n!}$.

Total Feature Attribution Equivalence for Model Fingerprint and SHAP

To compare model fingerprint with SHAP, we must compute an aggregation of the model fingerprint logical components for feature a . Its share of an interaction effect is $1/t$ where $t = |T|$ is

the order of the interaction. For example, feature a would include 1/3 of each third-order interaction effect that includes it, with the other two features in each triplet claiming the other 2/3 collectively.

Definition 3 (Model fingerprint feature-level summary). We summarize the model fingerprint importance of feature a by aggregating its share of all effects that include it, $T \ni a$.

$$\phi_a^{MF} \equiv \begin{cases} \sum_{T \ni a} \frac{1}{t} \psi_T, & a \in N \\ \mathbb{E}[\hat{f}(x)], & a = \emptyset \end{cases} \quad (16)$$

Substituting from equation 6 for ψ_T , we have:

$$\phi_a^{MF} \equiv \begin{cases} \sum_{U \subseteq N} c_a^{MF}(U) \hat{f}_U, & a \in N \\ \mathbb{E}[\hat{f}(x)], & a = \emptyset \end{cases} \quad (17)$$

$$c_a^{MF}(U) = \sum_{\substack{T \supseteq U \\ T \ni a}} \frac{1}{t} (-1)^{t-u} \quad (18)$$

Theorem 1 (Equivalence of MF and SHAP feature-level attribution). The total attribution of feature a to a prediction function $\hat{f}(X)$ is identical for model fingerprint and SHAP counting methods.

$$\phi_a^{MF} = \phi_a^{SHAP} \quad (19)$$

See proof in the Appendix A. We also include a simple empirical illustration of the equivalent calculations in Appendix B.

Low-Order Approximation (LOA) of Model Fingerprint

Owing to its equivalence, the full explicit calculation of the model fingerprint feature attribution has the same exponential computation complexity as SHAP, which is often impractical. However, the bottom-up counting of model fingerprint allows us to introduce a method of Low-Order Approximation (LOA).

The appeal of the LOA rests on the insight that the dominant explanatory power for many predictive models comes from low-order effects—such as first-order and second-order

interactions—while higher-order interactions contribute relatively little and are also difficult to estimate with precision due to data sparsity. The LOA method focuses on these dominant low-order terms and aggregates all remaining high-order variation into a single residual component, which can be distributed uniformly across variables if a summary feature attribution value is needed. This strategy reduces the computational burden from exponential to roughly polynomial time while preserving the essential properties of the original model fingerprint components.

Definition 4 (Residual higher order effects). The higher order effects beyond a chosen maximum of p is the residual of \hat{f} after subtracting all lower-order components:

$$\psi_{N|p} \equiv \hat{f} - \sum_{\substack{T \subseteq N \\ |T| \leq p}} \psi_T \quad (20)$$

This approximation reduces computation time from exponential to polynomial levels, as the model fingerprint efficiently computes lower-order effects. Nevertheless, aggregating high-order effects breaks the equivalence of model fingerprint with SHAP, and the resulting aggregate feature attribution for model fingerprint no longer preserves the dummy property of SHAP. Nevertheless, the LOA is often well justified in practice, and one may estimate the fraction of predictive variance explained by lower-order components to gauge the degree of approximation throughout the process. Consequently, while the imputed higher-order effects may allocate some nonzero value to aggregate feature attribution to truly null effect features in violation of the dummy axiom, in practice the LOA fingerprint often assigns only a minimal value to these features. Moreover, the explicitly computed lower-order effects are not distorted, the framework remains model agnostic, and the other three SHAP axioms are preserved.

Practical Considerations and Comparison of Approaches

In this section, we first outline several common SHAP methods that aim to reduce practical computation time. We also discuss the empirical estimation of conditional expectations, which can be computationally demanding and statistically delicate, especially in high-dimensional settings. We discuss how estimation methods affect model interpretability and results, and we compare the various SHAP methods with the model fingerprint method and its LOA, evaluating their respective strengths and limitations.

SHAP Practical Methods

Adjacent to the core SHAP method there exists a family of practical algorithms that approximate exact SHAP values under different modeling assumptions and data constraints. The most prominent variants include:

- **Linear SHAP:** Designed for linear or generalized linear models, this method leverages the closed-form solution of Shapley values in additive models. By exploiting the linear structure, it computes exact contributions with minimal computational cost. However, it assumes all features are independent.
- **Kernel SHAP:** A model-agnostic approach that estimates SHAP values through weighted linear regression over randomly sampled feature coalitions. Although flexible and broadly applicable, Kernel SHAP scales exponentially with the number of features and requires careful sampling to balance accuracy and efficiency.
- **Deep SHAP:** Tailored for deep neural networks, Deep SHAP combines ideas from DeepLIFT and SHAP. It propagates contribution scores through the network layers, greatly improving computational efficiency while maintaining theoretical guarantees for specific activation functions.
- **Tree SHAP:** Optimized for tree-based models such as gradient-boosted trees or random forests. Tree SHAP exploits the tree structure to compute exact SHAP values in polynomial time, making it the standard choice for many ensemble methods.

Estimation of Conditional Expectation

Despite the introduction of various practical SHAP variants and the model fingerprint LOA, all of these methods still face a common and fundamental challenge: estimating conditional expectations. There are two main popular approaches: observational conditional expectation and interventional conditional expectation. Chen et al. (2020) discuss the distinct advantages and limitations of these two methods for SHAP.

- **Observational Conditional Expectation:**

The observational conditional expectation is a method to compute the expectation directly from the observed values in a dataset, in which we set the value function $v(S) = \mathbb{E}[f(X)|X_S = x_S]$. In the calculation, the conditional distribution of the remaining features is taken exactly as it appears in the dataset conditional on $X_S = x_S$, which means the calculation relies solely on the patterns present in the empirical data sample.

This method preserves the statistical relationships present in a dataset and maintains fidelity to the data-generating process. However, it suffers from several drawbacks:

- *Continuity and sparsity*: When features are continuous, the data points are often too sparse to reliably compute expectations, which increases the variance of estimates.
- *Restricted scenario*: It can only evaluate values present in the dataset and cannot extrapolate to unseen values.
- *Attribution to irrelevant features*: As noted by Janzing et al. (2019) and Sundararajan & Najmi (2019), this approach may allocate non-zero contributions to irrelevant variables, violating the dummy property.

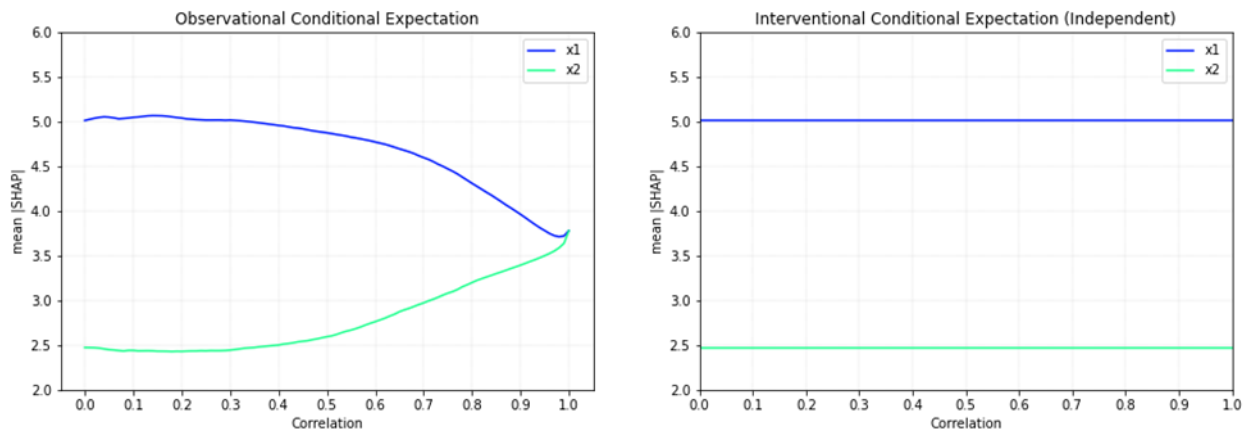
- **Interventional Conditional Expectation:**

An alternative is the interventional conditional expectation, in which we define the value function as $v(S) = \mathbb{E}[f(X)|do(X_S = x_S)]$. In principle, this method tries to calculate the expectation by sampling or mimicking the distribution of input variables. In theory, it can overcome some of the limitations of the observational method such as its inability to handle unseen values. However, these methods are computationally expensive, difficult to scale, and often rely heavily on the observed data as well as specified assumptions. As a result, they cannot perfectly reproduce the underlying distribution and may overfit to the existing dataset.

To simplify, a more common practice is to directly break the correlations among features and impose an independence assumption by defining the value function as $v(S) = \mathbb{E}[f(X \setminus X_S, X_S = x_S)]$. Many practical SHAP such as linear SHAP use this method. The independence assumption allows conditional expectations to be consistently computed even for values outside the dataset. It ensures tractability, avoids reliance on sparse or incomplete data, and guarantees that conditional expectations are always defined. Nevertheless, it comes with trade-offs that by discarding correlations, it may offer a misleading sense of attribution in settings with highly overlapping information across features.

Consider a simple model $f(x_1, x_2) = 2x_1 + x_2$ as an illustration of the difference between these two methods. First, we generate 10000 samples of each variable between $[0,10]$ with varying correlation levels. Then for each sample, we compute their SHAP values and aggregate them into a global measure as the average of absolute SHAP values. This metric reflects the average importance of each variable across the entire dataset and is able to represent the contributions of variables.

Exhibit 5: Comparison between Observational and Interventional Conditional Expectation



In Exhibit 5, the blue line shows the contribution of x_1 , and the green line shows the contribution of x_2 . When x_1 and x_2 are independent, the global contribution of average $|\text{SHAP}|$ is at 5 for x_1 and 2.5 for x_2 , which has the same ratio as the coefficients. However, under the observational expectation, as correlation between x_1 and x_2 increases, the contributions gradually converge toward equality. Meanwhile, the interventional approach would preserve the constant contributions, independent of correlation.

Both approaches are reasonable, but they address different interpretability questions. With the observational expectation, it makes sense to assign equal importance to the two features as their correlation grows since they become nearly indistinguishable. On the other hand, the interventional method is also valid because it directly reflects the model's coefficients, staying true to its structure. This comparison highlights a fundamental choice: whether to be faithful to the data (observational expectation) or faithful to the prediction function or model (interventional expectation). The former preserves dataset relationships but struggles with sparse or unseen values, while the latter ensures functional fidelity but sacrifices correlation structure.

Practical Method Comparison

Exhibit 6 presents a concise overview of the main strengths and limitations of these approaches. The model fingerprint LOA is the only model-agnostic approach that can provide substantial model interpretation with less than exponential compute time.

Exhibit 6: Comparison of Approaches

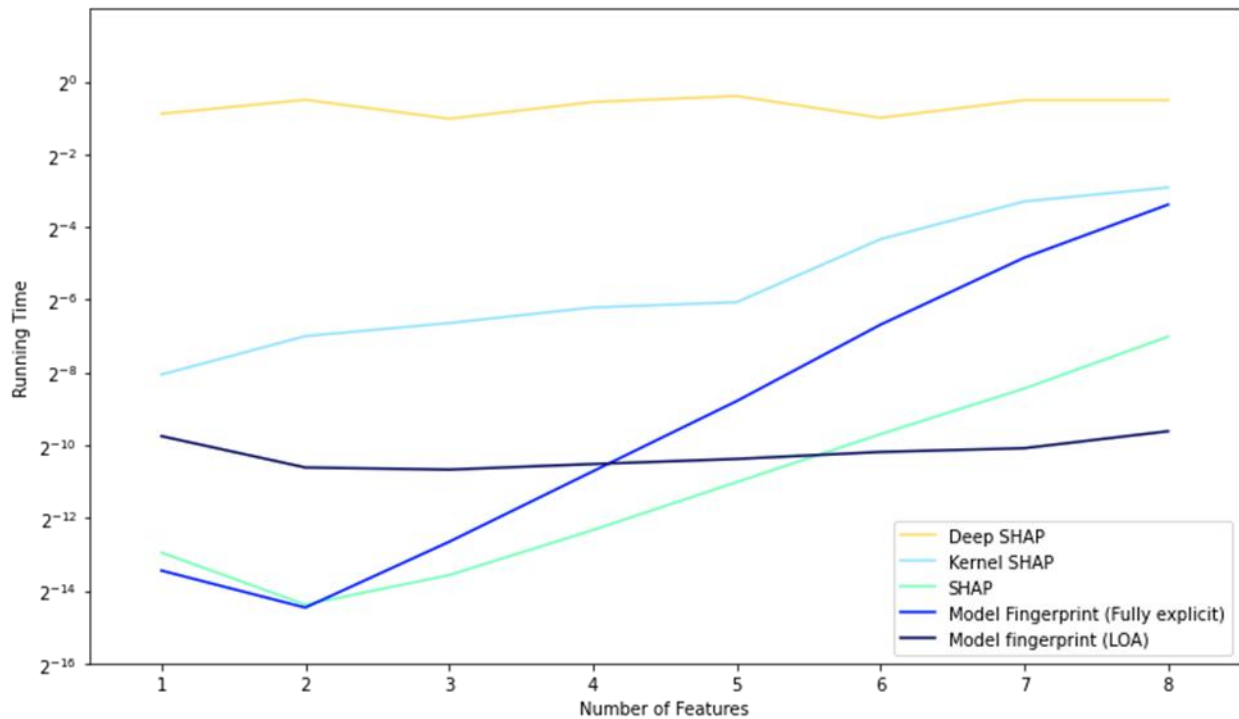
Method	No Independence Assumption Required	Model Agnostic	Shapley Properties	Computation Time
SHAP	✓	✓	✓	Exponential
Model fingerprint (Fully explicit)	✓	✓	✓	Exponential
Model fingerprint (LOA)	✓	✓	Violate Dummy	Polynomial
KernelSHAP	×	✓	✓	Exponential
LinearSHAP	×	×	✓	Constant
DeepSHAP	×	×	✓	Polynomial
TreeSHAP	✓	×	✓	Polynomial

Real-World Illustration

We demonstrate the efficiency of the model fingerprint LOA using the California Housing Price dataset which comprises eight variables. To thoroughly assess interpretability methods, we train eight neural network models, each incorporating an incremental number of input features: the first model uses only the first feature, the second uses first two, and so on, until the eighth model includes all eight features. We then apply a range of interpretability techniques to each model.

In Exhibit 7, the horizontal axis denotes the number of model input features, while the vertical axis depicts runtime on a logarithmic scale. Both Kernel SHAP, SHAP, and the fully-explicit model fingerprint exhibit exponential increases in computation time as the feature count rises. In contrast, Deep SHAP and the model fingerprint LOA demonstrate polynomial scaling.

Exhibit 7: Running Time of Different Methods

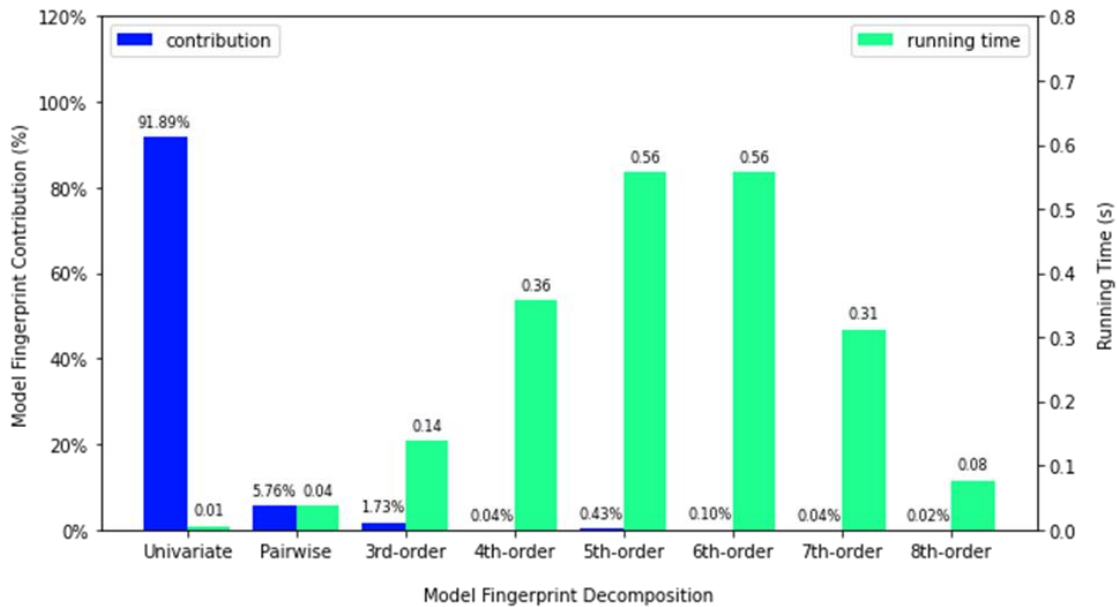


Notably, when the number of features is small (less than four in this example), the LOA method shows slightly slower running time compared to fully-explicit model fingerprint and SHAP. This is because the fully-explicit model fingerprint and SHAP perform direct additive calculations, whereas the LOA method requires an additional step: computing higher-order contributions by referencing previously calculated values and subtracting them from the prediction output. This is also the reason why computing time of fully-explicit model fingerprint is a bit longer than SHAP. When the number of variables is small, this extra step introduces additional runtime. However, when a model only has a small number of variables, computational complexity is typically of little to no concern. In common machine learning applications, practitioners deal with tens, or even hundreds of input variables. When the number of input variables is large, the divergence in computation time will become much more dramatic, and the efficiency gains more pronounced.

The California Housing Price dataset illustrates a clear relationship between computational cost and the proportion of prediction explained using the LOA approximation method. As depicted in Exhibit 8, each order of effect—univariate, pairwise, third-order, and beyond—offers insight into both its fractional contribution to explanatory variance and the time required for computation. Remarkably, lower-order effects, particularly those involving independent features and pairwise interactions, account for about 97% of the total prediction. Higher-order interactions add only a

small fraction to the interpretive power while demanding significantly greater computational resources. While this contrast is pronounced even in this relatively small illustration with fewer than ten features, the gap in computation time will be orders of magnitude greater in settings that involve tens or hundreds of input variables.

Exhibit 8: Explanatory Contribution versus Computation Cost



Conclusion

The model fingerprint framework stands out as both theoretically robust and practically effective model-agnostic method for interpreting prediction logic. It delivers comprehensive insights both at the level of individual predictions and for a prediction function (model) overall. Notably, the aggregate feature attribution values computed by model fingerprint for a given prediction input are mathematically equivalent to those from the SHAP method, thereby inheriting all of the desirable Shapley properties.

One of the key advantages of model fingerprint lies in its ability to explicitly dissect and illuminate the logical components underlying a prediction: linear effects, nonlinear effects, pairwise interactions, and higher-order interactions. Unlike standard SHAP implementations, which do not directly offer this level of transparency, model fingerprint enables users to distinguish features that exert a straightforward linear influence from those whose primary role is to condition the

effects of others. This distinction aids meaningful interpretation and provides a deeper understanding of how a prediction routine behaves.

The low-order approximation (LOA) of model fingerprint offers a marked reduction in computational complexity, operating in polynomial time in contrast to the exponential time of alternatives such as Deep SHAP and Kernel SHAP. This notable efficiency makes it well-suited for interpreting even highly complex models, including deep neural networks.

Appendix A: Proof of Theorem 1 (Equivalence of MF and SHAP feature-level attribution)

Our goal is to show that $\phi_a^{MF} = \phi_a^{SHAP}$, and it is obvious when a is \emptyset . Then when $a \in N$, it suffices to show that the coefficients in equations 14 and 17 are identical, $c_a^{MF}(U) = c_a^{SHAP}(U)$ for any U . In other words, model fingerprint and SHAP count the same components, but they do so in a different sequence according to their canonical definitions.

Recall the definition of model fingerprint coefficients from equation 18:

$$c_a^{MF}(U) = \sum_{\substack{T \supseteq U \\ T \ni a}} \frac{1}{t} (-1)^{t-u}$$

Consider first the case where $a \in U$. Then $T \supseteq U$ and for the larger set T , consider the number of elements as $t = u + r$ where $r \in \{0, \dots, n - u\}$. A given choice of r introduces $\binom{n-u}{r}$ supersets. Therefore:

$$c_a^{MF}(U) = \sum_{r=0}^{n-u} \binom{n-u}{r} \frac{(-1)^r}{u+r}$$

For clarity, define the following key sum which depends only on the number of elements u in U :

$$H_M(u) = \sum_{r=0}^M \binom{M}{r} \frac{(-1)^r}{u+r}$$

Here $M = n - u$, so $c_a^{MF}(U) = H_{n-u}(u)$.

Now, consider the case where $a \notin U$. In this case, T must contain $U \cup \{a\}$, which has $u + 1$ elements. Similar to before, write $t = u + 1 + r$ with $r \in \{0, \dots, n - u - 1\}$. Now there are $\binom{n-u-1}{r}$ supersets, and we have:

$$c_a^{MF}(U) = \sum_{r=0}^{n-u-1} \binom{n-u-1}{r} \frac{(-1)^{r+1}}{u+r+1} = -H_{n-u-1}(u+1)$$

We have written the coefficients exclusive in terms of $H_M(u)$, so we can focus on this function. We will show that $H_M(u)$ exhibits a recurrence that allows for argument by induction.

Pascal's identity holds that $\binom{M+1}{r} = \binom{M}{r} + \binom{M}{r-1}$. Then:

$$H_{M+1}(u) = \sum_{r=0}^{M+1} \binom{M+1}{r} \frac{(-1)^r}{u+r}$$

$$H_{M+1}(u) = \sum_{r=0}^M \binom{M}{r} \frac{(-1)^r}{u+r} + \sum_{r=1}^{M+1} \binom{M}{r-1} \frac{(-1)^r}{u+r}$$

Shift $k = r - 1$ in the second term, so:

$$H_{M+1}(u) = \sum_{r=0}^M \binom{M}{r} \frac{(-1)^r}{u+r} - \sum_{k=0}^M \binom{M}{k} \frac{(-1)^k}{u+1+k}$$

Thus, we have a recurrence relation for H :

$$H_{M+1}(u) = H_M(u) - H_M(u+1)$$

We show by induction that the closed form expression $H_M(u) = \frac{M!(u-1)!}{(u+M)!}$ satisfies the base and recurrence conditions. For $M = 0$, $H_0(u) = \frac{1}{u}$. Now, evaluate $H_{M+1}(u)$ as:

$$H_{M+1}(u) = H_M(u) - H_M(u+1)$$

$$H_{M+1}(u) = \frac{M!(u-1)!}{(u+M)!} - \frac{M!u!}{(u+M+1)!}$$

$$H_{M+1}(u) = \frac{M!(u-1)!}{(u+M+1)!} ((u+M+1) - u)$$

$$H_{M+1}(u) = \frac{(M+1)!(u-1)!}{(u+M+1)!}$$

Finally, we plug in this closed form expression for the model fingerprint formulas for the coefficient to show their equivalence to the SHAP coefficients in equation 15 for both cases.

If $a \in U$:

$$c_a^{MF}(U) = H_{n-u}(u) = \frac{(n-u)!(u-1)!}{n!} = c_a^{SHAP}(U)$$

If $a \notin U$:

$$c_a^{MF}(U) = -H_{n-u-1}(u+1) = -\frac{(n-u-1)!u!}{n!} = c_a^{SHAP}(U)$$

Q.E.D.

Appendix B: Simple Illustration of MF and SHAP Calculations

We illustrate model fingerprint and SHAP calculations for a discrete empirical example that contains redundancy in feature values to make the estimation of conditional expectation values straightforward.

Exhibit B1: Illustration of SHAP Calculations

Observation	x_A	x_B	x_C	\hat{f}
(1)	0	1	6	1
(2)	-1	1	6	-1
(3)	3	1	6	7
(4)	7	1	6	15
(5)	3	2	6	8
(6)	5	2	6	12

Model Fingerprint Calculation

Begin by calculating the partial predictions for $x = (3,2,6)$.

$$\hat{f}_A(3) = \frac{15}{2} - 7 = \frac{1}{2}$$

$$\hat{f}_B(2) = 10 - 7 = 3$$

$$\hat{f}_C(6) = 6 - 6 = 0$$

$$\hat{f}_{A,B}(3,2) = 8 - 7 = 1$$

$$\hat{f}_{A,C}(3,6) = \frac{15}{2} - 7 = \frac{1}{2}$$

$$\hat{f}_{B,C}(2,6) = 10 - 7 = 3$$

$$\hat{f}_{A,B,C}(3,2,6) = 8 - 7 = 1$$

Next, calculate the fingerprint components incremental to all prior effects.

$$\psi_A(3) = \hat{f}_A(3) = \frac{1}{2}$$

$$\psi_B(2) = \hat{f}_B(2) = 3$$

$$\psi_C(6) = \hat{f}_C(6) = 0$$

$$\psi_{A,B}(3,2) = \hat{f}_{A,B}(3,2) - (\psi_A(3) + \psi_B(2)) = 1 - \frac{1}{2} - 3 = -\frac{5}{2}$$

$$\psi_{A,C}(3,6) = \hat{f}_{A,C}(3,6) - (\psi_A(3) + \psi_C(6)) = \frac{1}{2} - \frac{1}{2} - 0 = 0$$

$$\psi_{B,C}(2,6) = \hat{f}_{B,C}(2,6) - (\psi_B(2) + \psi_C(6)) = 3 - 3 - 0 = 0$$

$$\begin{aligned} \psi_{A,B,C}(3,2,6) &= \hat{f}_{A,B,C}(3,2,6) - (\psi_{A,B}(3,2) + \psi_{A,C}(3,6) + \psi_{B,C}(2,6)) - (\psi_A(3) + \psi_B(2) + \psi_C(6)) \\ &= 1 + \frac{5}{2} - \frac{1}{2} - 3 = 0 \end{aligned}$$

Lastly, calculate each aggregate feature importance value.

$$\phi_A^{MF}(3,2,6) = \psi_A(3) + \frac{1}{2}(\psi_{A,B}(3,2) + \psi_{A,C}(3,6)) + \frac{1}{3}(\psi_{A,B,C}(3,2,6)) = \frac{1}{2} - \frac{5}{4} = -\frac{3}{4}$$

$$\phi_B^{MF}(3,2,6) = \psi_B(2) + \frac{1}{2}(\psi_{A,B}(3,2) + \psi_{B,C}(2,6)) + \frac{1}{3}(\psi_{A,B,C}(3,2,6)) = 3 - \frac{5}{4} = \frac{7}{4}$$

$$\phi_C^{MF}(3,2,6) = \psi_C(6) + \frac{1}{2}(\psi_{A,C}(3,6) + \psi_{B,C}(2,6)) + \frac{1}{3}(\psi_{A,B,C}(3,2,6)) = 0$$

$$\phi_\emptyset^{MF}(3,2,6) = \mathbb{E}[\hat{f}(x)] = \frac{1}{6}(1 - 1 + 7 + 15 + 8 + 12) = 7$$

These components, plus the baseline average prediction, sum to the total prediction.

$$\hat{f}(3,2,6) = \phi_A^{MF}(3,2,6) + \phi_B^{MF}(3,2,6) + \phi_C^{MF}(3,2,6) + \phi_\emptyset^{MF}(3,2,6) = -\frac{3}{4} + \frac{7}{4} + 0 + 7 = 8$$

SHAP Calculation

Begin with the same partial predictions as in the model fingerprint calculation for $x = (3,2,6)$, repeated here for convenience.

$$\hat{f}_A(3) = \frac{15}{2} - 7 = \frac{1}{2}$$

$$\hat{f}_B(2) = 10 - 7 = 3$$

$$\hat{f}_C(6) = 6 - 6 = 0$$

$$\hat{f}_{A,B}(3,2) = 8 - 7 = 1$$

$$\hat{f}_{A,C}(3,6) = \frac{15}{2} - 7 = \frac{1}{2}$$

$$\hat{f}_{B,C}(2,6) = 10 - 7 = 3$$

$$\hat{f}_{A,B,C}(3,2,6) = 8 - 7 = 1$$

Calculate the incremental affect of adding each feature to each cohort.

$$\hat{f}_A(3) - \hat{f}_\emptyset = \frac{1}{2}$$

$$\hat{f}_{A,B}(3,2) - \hat{f}_B(2) = 1 - 3 = -2$$

$$\hat{f}_{A,C}(3,6) - \hat{f}_C(6) = \frac{1}{2}$$

$$\hat{f}_{A,B,C}(3,2,6) - \hat{f}_{B,C}(2,6) = 1 - 3 = -2$$

$$\hat{f}_B(2) - \hat{f}_\emptyset = 3$$

$$\hat{f}_{A,B}(3,2) - \hat{f}_A(3) = 1 - \frac{1}{2} = \frac{1}{2}$$

$$\hat{f}_{B,C}(3,2) - \hat{f}_C(3) = 3$$

$$\hat{f}_{A,B,C}(3,2,6) - \hat{f}_{A,C}(3,6) = 1 - \frac{1}{2} = \frac{1}{2}$$

$$\hat{f}_C(6) - \hat{f}_\emptyset = 0$$

$$\hat{f}_{A,C}(3,6) - \hat{f}_A(3) = \frac{1}{2} - \frac{1}{2} = 0$$

$$\hat{f}_{B,C}(3,6) - \hat{f}_B(2) = \frac{1}{2} - \frac{1}{2} = 3 - 3 = 0$$

$$\hat{f}_{A,B,C}(3,2,6) - \hat{f}_{A,B}(3,2) = 1 - 1 = 0$$

Lastly, calculate each aggregate feature importance value.

$$\phi_A^{SHAP} = \frac{(1-1)!(3-1)!}{3!} \left(\frac{1}{2}\right) + \frac{(2-1)!(2-2)!}{3!} (-2) + \frac{(2-1)!(2-2)!}{3!} \left(\frac{1}{2}\right) + \frac{(3-1)!(3-2)!}{3!} (-2) = -\frac{3}{4}$$

$$\phi_B^{SHAP} = \frac{(1-1)!(3-1)!}{3!} (3) + \frac{(2-1)!(2-2)!}{3!} \left(\frac{1}{2}\right) + \frac{(2-1)!(2-2)!}{3!} (3) + \frac{(3-1)!(3-2)!}{3!} \left(\frac{1}{2}\right) = \frac{7}{4}$$

$$\phi_C^{SHAP} = \frac{(1-1)!(3-1)!}{3!} (0) + \frac{(2-1)!(2-2)!}{3!} (0) + \frac{(2-1)!(2-2)!}{3!} (0) + \frac{(3-1)!(3-2)!}{3!} (0) = 0$$

$$\phi_\emptyset^{SHAP}(3,2,6) = \mathbb{E}[\hat{f}(x)] = \frac{1}{6}(1 - 1 + 7 + 15 + 8 + 12) = 7$$

These components, plus the baseline average prediction, sum to the total prediction.

$$\hat{f}(3,2,6) = \phi_A^{SHAP}(3,2,6) + \phi_B^{SHAP}(3,2,6) + \phi_C^{SHAP}(3,2,6) + \phi_\emptyset^{SHAP}(3,2,6) = -\frac{3}{4} + \frac{7}{4} + 0 + 7 = 8$$

Notes

This material is for informational purposes only. The views expressed in this material are the views of the authors, are provided “as-is” at the time of first publication, are not intended for distribution to any person or entity in any jurisdiction where such distribution or use would be contrary to applicable law, and are not an offer or solicitation to buy or sell securities or any product. The views expressed do not necessarily represent the views of State Street Markets® and/or State Street Corporation® and its affiliates.

References

Li, Y., D. Turkington and A. Yazdani. 2020. “Beyond the Black Box: An Intuitive Approach to Investment Prediction with Machine Learning. *The Journal of Financial Data Science*, 2 (1).

Li, Y., Simon, Z. and Turkington, D., 2022. Investable and interpretable machine learning for equities. *The Journal of Financial Data Science*, 4(1), pp.54-74.

Friedman, J. H. 2001. “Greedy Function Approximation: A Gradient Boosting Machine.” *Annals of Statistics*, 29 (5).

Friedman, J.H. and Popescu, B.E., 2008. Predictive learning via rule ensembles.

Shapley, L.S., 1953. A value for n-person games.

Lundberg, S.M. and Lee, S.I., 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Lundberg, S.M., Erion, G.G. and Lee, S.I., 2018. Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888.

Chen, H., Janizek, J.D., Lundberg, S. and Lee, S.I., 2020. True to the model or true to the data?. arXiv preprint arXiv:2006.16234.

Janzing, D., Minorics, L. and Blöbaum, P., 2020, June. Feature relevance quantification in explainable AI: A causal problem. In *International Conference on artificial intelligence and statistics* (pp. 2907-2916). PMLR.

Sundararajan, M. and Najmi, A., 2020, November. The many Shapley values for model explanation. In *International conference on machine learning* (pp. 9269-9278). PMLR.