Interpreting Machine Learning Models: A Practical Guide for Applied Economists

Adam D. Rennhoff*

July 14, 2025

Abstract

This paper provides a practical guide for economists seeking to interpret machine learning models with the same statistical rigor expected in traditional econometric analysis. Using simulated data with known functional relationships, I demonstrate how to identify important features and understand their directional effects using model-agnostic interpretation tools. Three complementary approaches for measuring feature importance are examined: built-in measures, permutation importance, and SHAP-based methods. For understanding directional relationships, I show how partial dependence plots, individual conditional expectation (ICE) plots, and SHAP values with bootstrapped confidence intervals can reveal both average effects and heterogeneity across observations. These tools successfully recover the known nonlinear relationships in the simulated data, including quadratic terms, exponential functions, and interaction effects, without requiring researchers to specify functional forms a priori. The interpretation methods build naturally on familiar econometric concepts such as marginal effects, bootstrapping, and statistical significance testing, making them accessible to applied researchers. Rather than replacing traditional econometric reasoning, these tools extend it, particularly when relationships are nonlinear or unknown. This framework enables economists to harness the predictive power of machine learning while preserving the interpretability and inferential standards central to empirical research.

JEL Classification Codes: C45; C52

Keywords: Machine learning; applied econometrics; model interpretation; predictive modeling

*Department of Economics and Finance, Jennings A. Jones College of Business, Middle Tennessee State University, Murfreesboro, TN 37132. E-mail: Adam.Rennhoff@mtsu.edu. Phone: 615-898-2931.

1 Introduction

The use of machine learning techniques in academic research has increased across a wide variety of disciplines, including applications in medicine (Park et al. (2021)) and psychology (Gladstone et al. (2019)). Economics, as noted by Desai (2023), has also seen a significant increase in the use of machine learning in published academic research.

Within economics, the use of machine learning generally falls into one of two categories. The first category is what we might refer to as "input production." This refers to the use of machine learning techniques as an intermediate step in the research process. Examples include natural language processing to parse text (Gentzkow et al. (2019a) and Hansen et al. (2017)), inference based on visual or photographic data (Jean et al. (2016) and Gebru et al. (2017)), and K-means clustering to group sample observations (Owens and Rennhoff (2018)). In papers of this type, machine learning is used to produce an input, such as a variable that measures an individual's likely political orientation based on speech or writings (Gentzkow et al. (2019b)), which is ultimately incorporated into what we might consider a more "traditional" econometric approach, such as linear regression or a binary choice model.

The second general category involves using machine learning as a replacement for or complement to traditional causal inference methods, such as difference-in-differences.[1] Wager and Athey (2018) are credited with deriving the so-called *causal forest*, which is a modification of the random forest model that allows for the estimation of heterogeneous treatment effects. Two recent examples of applications using Wager and Athey's causal forest are Knittel and Stolper (2025), who examine the impact of a behavioral "nudge" on household electricity consumption, and Owens and Rennhoff (2025), who examine the impact of government-owned broadband networks on local internet network quality. While causal forests represent an important development in causal inference, there remains significant potential for machine learning in other econometric contexts.

It is worth noting that economists may already be familiar with some machine learning techniques that maintain the interpretability of traditional econometric models. LASSO and Ridge regression, for example, are regularization methods that extend the traditional linear regression model by adding penalty terms to prevent overfitting. While these methods are technically machine learning approaches, they retain the familiar structure of linear regression with interpretable β coefficients. However, this paper focuses on more complex, nonlinear machine learning models that may offer superior predictive performance but require different interpretation strategies. Unlike LASSO and Ridge, which are constrained to linear relationships and may not outperform OLS in terms of pure prediction accuracy, the methods discussed in this guide can capture complex nonlinear patterns and interactions that traditional econometric models might miss.

While many economists are comfortable interpreting regression coefficients and using hypothesis tests to assess significance, the outputs of machine learning models can seem opaque

[1] Interest in estimating causal effects has grown across economics. Using a large language model (LLM) and more than 44,000 NBER and CEPR working papers, Garg and Fetzer (2024) finds that almost 30% of current economics working papers claim to measure a causal effect.

by comparison. As these models become more common in applied research, economists increasingly need tools that bring similar interpretive clarity to machine learning outputs.

An underexplored but promising role for machine learning, and one that this paper addresses, is its use in settings where economists have traditionally relied on linear regressions or discrete choice models. Some readers may be familiar with the phrase "machine learning is about prediction; econometrics is about explanation," which echoes themes discussed in Breiman (2001b). This phrase emphasizes the perceived trade-off between those focused on prediction and those focused on interpretation and inference.[2] Given the strength of machine learning models at capturing complex non-linear relationships in data without requiring researchers to specify functional forms a priori, I argue that more economists should incorporate machine learning models into their analytical toolbox. This paper is intended to serve as a guide for researchers who are well-versed in point estimates and conventional hypothesis testing and who want to learn more about methods for conveying machine learning results to an academic audience. While this paper focuses on regression problems for clarity of exposition, the interpretation methods discussed are equally applicable to classification tasks. The paper is organized to specifically address two key questions: (a) which independent variables have an important statistical relationship with the dependent variable, and (b) how are the independent variables related to the dependent variable?

2 Methodology: Simulated Data and Model Implementation

This section presents the methodological framework used to demonstrate machine learning interpretation techniques throughout this paper. Rather than analyzing a specific research question with real-world data, I generate simulated data with known underlying relationships and train a machine learning model using this data. Using simulated data ensures that we know the true relationships between variables, which allows us to evaluate whether interpretation tools recover these relationships accurately,[3]

Before proceeding to a description of the data generation process, it is worth briefly pausing to discuss terminology. In economics, empirical models typically refer to "outcome" or "dependent variables" (y), and "covariates," "regressors," or "independent variables" (X). In machine learning, these are generally referred to as the *target* and *features*, respectively. Similarly, econometric models are "estimated," whereas machine learning models are *fitted*. Throughout the remainder of this paper, I adopt machine learning terminology, such as "features" and "target," to better align with the conventions found in libraries such as

[2]The distinction between prediction and inference is often overstated. Traditional econometric methods like ordinary least squares explicitly optimize for predictive accuracy by minimizing the sum of squared residuals, and maximum likelihood estimation similarly seeks to maximize the probability of observing the sample data. The key difference is often not whether prediction matters, but rather the relative emphasis placed on interpretability versus predictive performance.

[3]Python code to replicate all model outputs is provided in a public Github repo: `https://github.com/adrennhoff/ML4Econ`.

`sklearn` or `xgboost`.

2.1 Data Generation Process

I begin by drawing values for 15 different features, denoted $x_1, \ldots, x_{15}$. Each feature x_j is drawn *iid* from the standard normal distribution. For each observation, I also draw an independent error term $\varepsilon \sim \mathcal{N}(0, 0.25^2)$. In total, 1,000 observations are generated.

These draws are used to construct a target variable y according to the following relationship:

$$\begin{aligned} y = {} & 1.2 \cdot \sin(x_1) + x_2^2 + e^{-x_3} + x_4 \cdot x_5 \\ & + 1.5 \cdot \ln(|x_6| + 1) + 2 \cdot x_8 + x_5 \cdot x_9 \\ & - 1.75 \cdot x_9 + 1.5 \cdot \min(1, x_7) + \varepsilon \end{aligned} \tag{1}$$

Several aspects of the data-generating process are worth noting. First, six features $(x_{10}, \ldots, x_{15})$ are excluded from Equation (1) and serve as noise variables that should not influence the target. This allows us to test whether feature importance methods can correctly assign low relevance to variables that have no effect on the outcome. Second, the included features contribute through various functional forms: linear terms (x_8), nonlinear transformations $(\sin(x_1),\ x_2^2,\ e^{-x_3},\ \min(1, x_7))$, and interaction effects $(x_4 \cdot x_5,\ x_5 \cdot x_9)$. This design reflects the complexity commonly encountered in applied research and allows us to evaluate how well machine learning interpretation methods can identify both the relevant features and the nature of their relationships.

Having established the data-generating process, I next turn to the selection and training of a machine learning model capable of capturing these complex relationships.

2.2 Machine Learning Model Selection and Training

There are a wide variety of machine learning models, ranging from relatively simple approaches (K-nearest neighbors) to highly complex architectures (deep neural networks). For this demonstration, I focus on two popular tree-based ensemble methods: random forest and XGBoost. While I briefly compare both models, the interpretation examples that follow focus on the better-performing XGBoost model.

Tree-based models are built from a series of binary splitting rules that create transparent decision paths, making them more appealing to economists accustomed to understanding how models arrive at their conclusions.[4] In contrast, neural networks with hidden layers and complex weight matrices often appear as "black boxes" that provide little insight into the underlying decision-making process. For economists, who emphasize identification strategies

[4]Indeed, some machine learning libraries allow users to examine the individual trees and decision rules used in generating the model's final predictions.

and the interpretability of empirical relationships, tree-based models offer a more natural bridge between traditional econometric approaches and machine learning methods.

Both random forest and XGBoost are ensemble methods, meaning that their final predictions represent the aggregation of predictions across many individual decision trees. This ensemble approach typically yields better predictive performance than single trees while maintaining the transparency that makes tree-based models attractive to researchers focused on understanding the underlying model.

2.2.1 Hyperparameter Tuning

Unlike many traditional econometric models, machine learning models are often not optimized for performance "right out of the box." This contrasts with econometric approaches like OLS, where the objective function (minimizing the sum of squared residuals) directly determines the parameter estimates, or maximum likelihood estimation, where the optimization criterion is explicitly built into the estimation procedure.

In contrast, many machine learning models rely on hyperparameters, which are settings that govern how the model is trained but are not estimated from the data in the same way as coefficients are in OLS. Hyperparameters are model training settings, such as the number of trees, learning rate, or maximum tree depth, that must be specified in advance and can significantly influence model performance. For example, in a random forest, a hyperparameter might specify how many trees to include. These values must be chosen before estimating the model's parameters and can significantly affect performance.[5] As a result, researchers typically use data-driven methods to select hyperparameters that balance model fit and generalization to new data.

While the exact steps for tuning the hyperparameters are situation- and data-dependent, the general approach, referred to as K-fold cross-validation, is as follows:

1. **Choose a grid of candidate hyperparameter values.**
2. **Split the dataset into K folds (typically 5 or 10).** These are equally sized subsets of the data, used to compute out-of-sample prediction.
3. **For each candidate hyperparameter combination:**
    - Loop over the K folds:
        (i) Treat one fold as the *validation set* and the remaining $K - 1$ folds as the *training set*.
        (ii) Fit the model using the training set and the current hyperparameter value.
        (iii) Use the fitted model to predict outcomes in the validation set.
        (iv) Record the prediction error (e.g., mean squared error).

[5]The idea of selecting hyperparameters mirrors how prior specifications can shape model behavior in Bayesian methods, such as MCMC.

4. **Average the prediction errors across all K folds.** This gives an estimate of the out-of-sample performance for each hyperparameter value.

5. **Select the hyperparameter combination with the lowest average out-of-sample prediction error.** This is considered the hyperparameter value that best generalizes to unseen data.

When selecting among multiple model types, as I do here when comparing random forest and XGBoost models, it is common to implement an additional preprocessing step before tuning hyperparameters. The dataset is typically divided into a *training set* and *testing set*, with common split ratios being 75-25 or 80-20. In the K-fold cross-validation procedure described above, only the training set would be used in Step 2.

Using an 80-20 train-test split, I tune both models (i.e., search for the optimal hyperparameters) and evaluate their performance.[6] The XGBoost model achieves a mean squared error of 1.5535 on the test set, compared to 3.9526 for the random forest model.[7] Given this performance advantage, all subsequent interpretation examples use the XGBoost model.

3 Feature Importance: Identifying Which Variables Matter

In this section, I focus on answering the first question posed in the introduction. Specifically, how can we identify and quantify those features having an important predictive relationship with our target variable? Below, I outline three different measures of what is referred to as feature importance.

3.1 Built-in Feature Importance Measures

Tree-based machine learning methods, such as the XGBoost model used in this demonstration, have what we might call a built-in measure of feature importance.[8] As the name suggests, feature importance ranks features according to their importance, which we will define below. In tree-based models like XGBoost, importance is most commonly measured using two metrics: "gain" and "weight." The term "gain" refers to the improvement in the model's performance, such as a reduction in mean squared error, when a feature is used to split the data. "Weight," on the other hand, refers to how often a feature is used across all the trees in the model, regardless of the size of its impact. Feature importances are often

[6] All specific details, including the selected hyperparameter grids, can be found in the Github repository.

[7] For comparison, an OLS model estimated on the training data and evaluated on the test data yields an MSE of 6.0536. Given the nonlinear data-generating process, the superior performance of machine learning models is expected.

[8] The phrase "built-in" indicates that it is an attribute computed automatically whenever a tree-based model is fit. In this regard, it is similar to an R^2 or a log-likelihood, both of which are returned without requiring any additional steps on the part of the researcher.

presented using horizontal bar charts, with features shown in descending order of importance. For our simulated data and the optimal XGBoost model, this feature importance plot is shown in Figure 1.

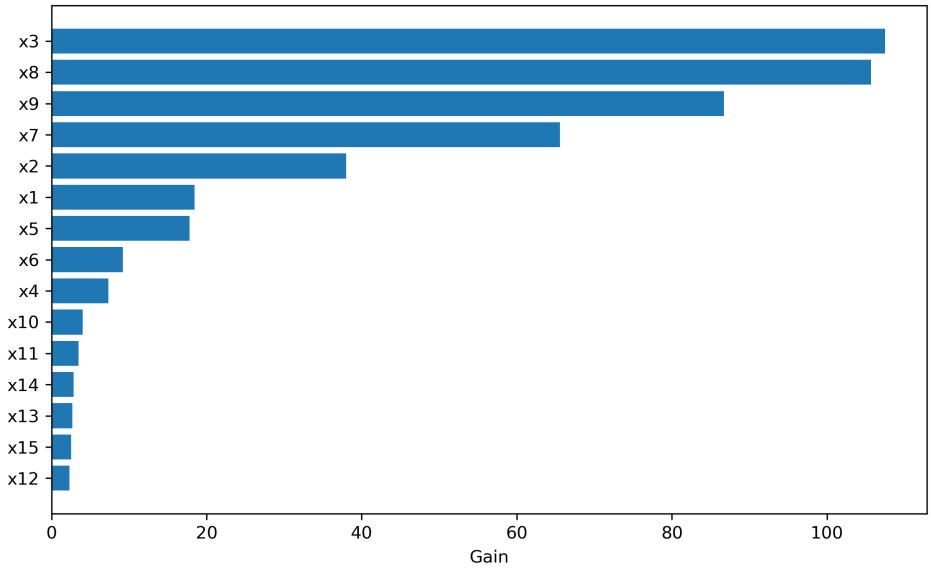

Figure 1: Feature Importance Based on Gain

As discussed above, there are multiple ways to define importance, with most libraries defaulting to "gain," the measure of how much a feature improves the model's performance when used in a split. Alternatives such as "weight," which counts how often a feature is used across all trees, can also be informative in some contexts. For example, a researcher may prefer weight when interested in how frequently the model relies on a feature, regardless of the size of its marginal effect.

From Figure 1, we see that x_3 and x_8 provide the largest reductions in mean squared error, followed closely by x_9. The six noise features (x_{10} through x_{15}) have small importances near zero, although not quite zero. These results align well with our known data-generating process from Equation (1). The model correctly identifies x_3 (which enters as e^{-x_3}) and x_8 (which enters linearly with coefficient 2.0) as highly important features. The prominence of x_9 also makes sense given its dual role in the interaction term $x_5 \cdot x_9$ and the linear term $-1.75 \cdot x_9$.

While built-in feature importance measures offer computational efficiency and ease of interpretation, they have several limitations. First, they are model-specific and may not generalize across different algorithms. Second, they can be biased toward features with more levels or those that appear earlier in the tree-building process. Finally, they may not capture complex interactions or provide uncertainty estimates around importance rankings. Given these limitations, researchers often complement built-in importance measures with model-agnostic approaches, such as permutation importance, which we examine next.

3.2 Permutation Importance

Given the limitations of built-in importance measures discussed above, researchers often complement these with model-agnostic approaches. Permutation importance, first introduced by Breiman (2001a) in the context of random forests, addresses several key concerns with built-in measures. Unlike gain-based importance, permutation importance is not model-dependent (i.e., it can be used with all supervised learning models), provides a more intuitive interpretation, and can be extended to include uncertainty estimates through bootstrapping.

Permutation importance is based on the conceptual idea that our trained model would generate worse predictions if we arbitrarily shuffled the values for an important feature. The algorithm works as follows: first, establish baseline performance by evaluating the trained model on the dataset.[9] Then, for each feature sequentially, randomly shuffle that feature's values across all observations (holding all other features at their observed values) and recompute the performance measure. The permutation importance is calculated as the difference between the shuffled performance and the baseline performance. Features that cause large performance degradation when shuffled are considered more important.

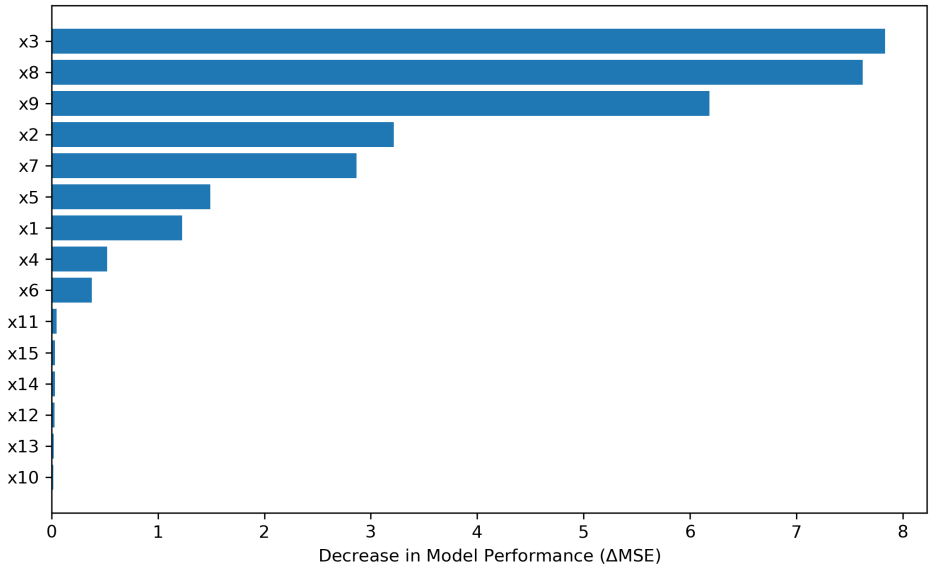

Figure 2: Permutation Feature Importance

Figure 2 displays a sorted horizontal bar chart, based on permutation feature importances. The results are quite similar to those in Figure 1, with x_3, x_8, and x_9 being the three most important features, in that order. Again, we see that our noise features are at the bottom of the list, with permutation importance showing them even closer to zero.

As economists, we are accustomed to using tools like p-values and confidence intervals to determine whether a relationship is statistically distinguishable from zero. Permutation importance accommodates this preference quite naturally. Rather than relying on a single

[9]Given that this is a "regression problem," I use mean squared error as the performance measure.

permutation (as in Figure 2), we can repeat the permutation process many times, treating each round as a bootstrap iteration. By computing the distribution of importance values for each feature across these iterations, we can construct bootstrapped confidence intervals, which provides a measure of uncertainty that aligns with familiar econometric practice.

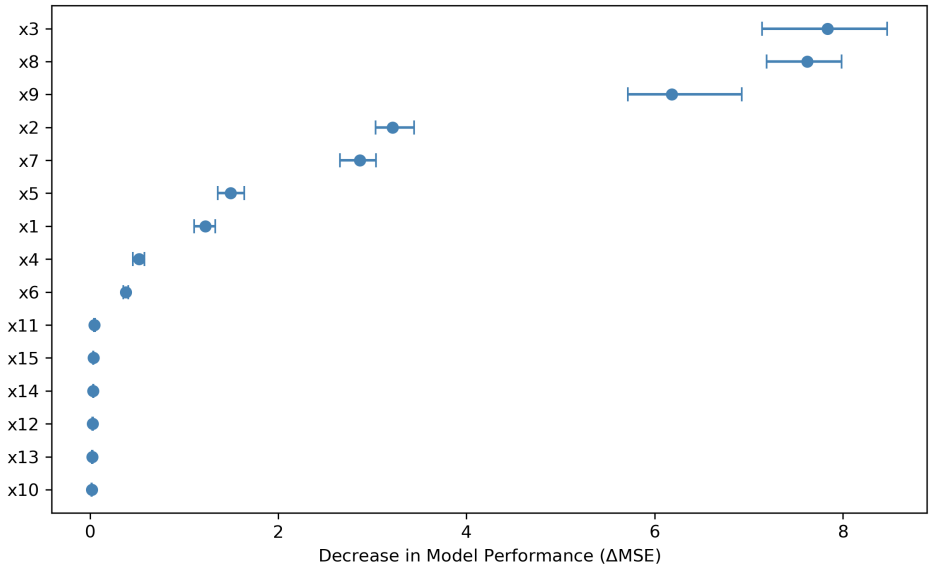

Figure 3: Permutation Importance with 95% Confidence Intervals

To assess the statistical uncertainty in these estimates, we extend the permutation process using bootstrapping. Figure 3 is a dot and whisker plot, showing the bootstrapped mean permutation importance (dot) and 95% confidence interval (whiskers) for our model features. The results are unsurprising, although it does illustrate how confident we can be that our noise features have virtually zero impact on model prediction. The same information could instead be organized as a table, more closely resembling a results table with a point estimate and confidence interval (or standard error/deviation).

While permutation importance provides valuable model-agnostic insights with uncertainty quantification, it still treats each feature in isolation. In the next subsection, we explore SHAP-based importance measures, which account for feature interactions and provide a unified framework connecting feature importance to individual prediction explanations.

3.3 SHAP-based Feature Importance

Our third measure of feature importance is based on SHAP (SHapley Additive exPlanations) values, which provide a theoretically grounded approach to understanding individual feature contributions. SHAP values adapt Shapley values from cooperative game theory, where the goal is to fairly allocate payouts among players based on their marginal contributions to all possible coalitions. In the classic example, if two firms form a joint venture, Shapley values determine how to fairly split the surplus by averaging each firm's marginal contribution across all possible joining scenarios.

In machine learning, this concept translates naturally: each feature becomes a "player" in a cooperative game where the objective is to explain a model's prediction for a specific observation. The SHAP value for a given feature represents its average marginal contribution to that prediction, computed across all possible combinations (coalitions) of the remaining features. SHAP values have several desirable properties: efficiency (SHAP values sum to the difference between the prediction and baseline), symmetry (features with identical contributions receive identical SHAP values), dummy (irrelevant features receive zero SHAP values), and additivity (the method consistently aggregates across multiple models).

SHAP values provide an additive decomposition of each model prediction, similar to how coefficients decompose predictions in linear regression. Importantly, SHAP values are observation-specific, meaning each data point has its own set of SHAP values that reflect how that observation's particular feature values contribute to its prediction.[10] This mirrors the concept of marginal effects in nonlinear models like logit or probit, where the effect of a one-unit change depends on the values of all other variables.[11]

While the exact computation of SHAP values can be computationally intensive, efficient approximation algorithms have been developed specifically for tree-based models such as random forests and XGBoost. A simplified numerical example of SHAP value calculation is provided in Appendix A.1.

For feature importance analysis, we aggregate individual SHAP values using the mean absolute SHAP value across all observations. This measure captures which features contribute most strongly to model predictions, regardless of direction. Formally, if $\phi_j^{(i)}$ represents the SHAP value for feature j in observation i, then the SHAP-based importance for feature j is calculated as the mean absolute SHAP value::

$$\text{SHAP Importance}_j = \frac{1}{n}\sum_{i=1}^{n} |\phi_j^{(i)}| \tag{2}$$

Figure 4 presents the SHAP-based feature importance rankings for our XGBoost model. The results closely align with our previous methods, again identifying x_3, x_8, and x_9 as the most important features. However, SHAP-based importance offers a key advantage: it naturally accounts for feature interactions when attributing importance, providing a more nuanced view of each feature's contribution that will prove valuable when we examine directional relationships in the following section.

Across all three methods, we consistently identify x_3, x_8, x_9, x_2, and x_7 as most important, with noise variables appropriately ranked near zero. This convergence gives us confidence in proceeding to examine how these important features influence predictions.

[10]While this section focuses on aggregated SHAP-based feature importance, SHAP values can also be visualized at the individual observation level using beeswarm plots, waterfall plots, and force plots. Examples of these visualization techniques are provided in Appendix A.2

[11]However, unlike marginal effects in logit and probit models, which maintain consistent directional relationships (e.g., always positive or always negative), SHAP values can vary in both magnitude and sign across observations, reflecting the complex nonlinear relationships captured by machine learning models.

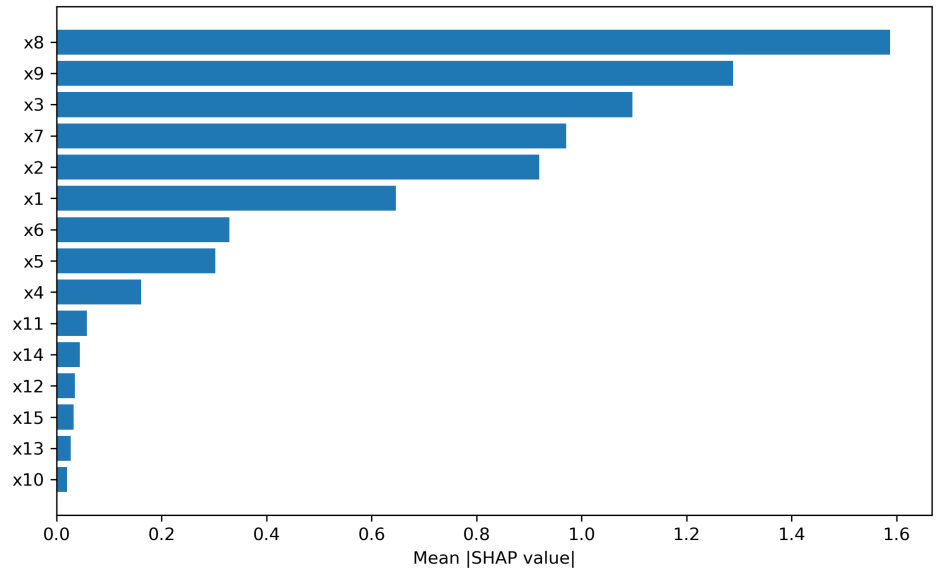

Figure 4: SHAP-Based Feature Importance

We return to observation-level SHAP visualizations, including beeswarm and waterfall plots, in Appendix A.2.

4 Directional Influence and Magnitude

4.1 Partial Dependence and ICE Plots

Having established which features are most important for our model's predictions, we now turn to understanding how these features influence the target variable. This section introduces two related visualization techniques that help economists understand the directional relationships captured by machine learning models: partial dependence plots (PDPs) and individual conditional expectation (ICE) plots.

4.1.1 Partial Dependence Plots

Partial dependence plots visualize the marginal effect of a feature on the predicted outcome, averaging over the values of all other features. In essence, PDPs answer the question: "On average, how does the prediction change as we vary one feature while holding other features constant?" This approach parallels the average marginal effect (AME) in nonlinear econometric models, where marginal effects are calculated for each observation and then averaged, rather than calculating a single marginal effect at the sample means.

The partial dependence function for feature x_j is defined as:

$$PD_j(x_j) = \mathbb{E}_{X_{-j}}\left[f(x_j, X_{-j})\right] = \frac{1}{n}\sum_{i=1}^{n} f(x_j, x_{-j}^{(i)}) \tag{3}$$

where $f(\cdot)$ represents our fitted XGBoost model, X_{-j} denotes all features except x_j, and $x_{-j}^{(i)}$ represents the observed values of all other features for observation i. This function represents the average prediction we would get by varying x_j while keeping all other features fixed at their observed values. To construct a PDP, we select a grid of values for feature x_j. For each grid point, we replace the observed values of x_j for all observations with that grid value, generate predictions using our trained model, and then average these predictions across all observations.[12]

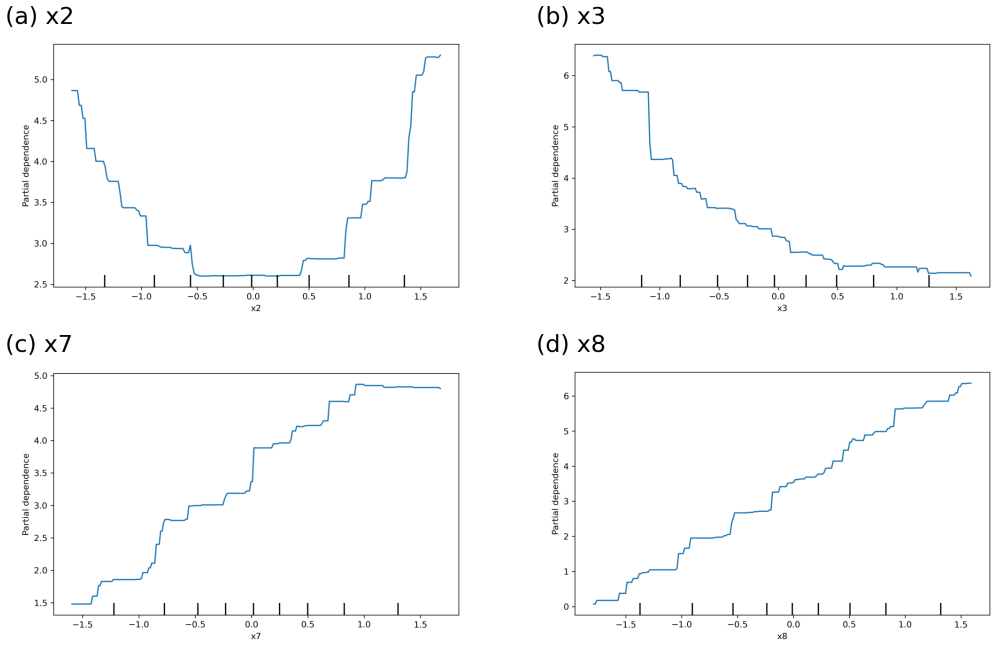

Figure 5: Partial Dependence Plots

Figure 5 displays partial dependence plots for four of our most important features: x_2, x_3, x_7, and x_8. These plots reveal the functional forms that our XGBoost model has learned from the data, which we can compare to the known relationships from our data-generating process in Equation (1).

The PDP for x_2 shows a clear quadratic relationship, with the prediction increasing as x_2 moves away from zero in either direction. This closely matches the x_2^2 term used when simulating the target data. For x_3, we see a downward-sloping, curved relationship rather than a linear one. This is consistent with the e^{-x_3} term used when simulating the target data.

[12]Both R and Python have built-in functions that perform these steps for you. It is, nevertheless, helpful to understand the steps performed by these functions.

The PDP for x_7 exhibits a distinctive kinked relationship. The plot increases, approximately linearly, for increasing values of x_7, but flattens out once x_7 exceeds 1. This captures the $\min(1, x_7)$ function from our data simulation, where the relationship is bounded above at 1. Finally, x_8 shows a clear linear relationship, consistent with its linear coefficient of 2.0 in Equation (1).

These results demonstrate that our XGBoost model has successfully learned the complex nonlinear relationships in our data without requiring us to specify the functional forms a priori, which is a key advantage of machine learning approaches over traditional econometric methods.

4.1.2 Individual Conditional Expectation (ICE) Plots

While partial dependence plots provide valuable insights into average relationships, they can mask important heterogeneity across observations. Individual conditional expectation (ICE) plots, introduced by Goldstein et al. (2015), address this limitation by showing how the prediction changes for each individual observation as we vary a single feature.

An ICE plot displays the function $f_i(x_j) = f(x_j, x_{-j}^{(i)})$ for each observation i, where we vary x_j over a grid of values while holding all other features at their observed values for that specific observation. In this regard, a PDP represents the average of all the individual ICE curves.

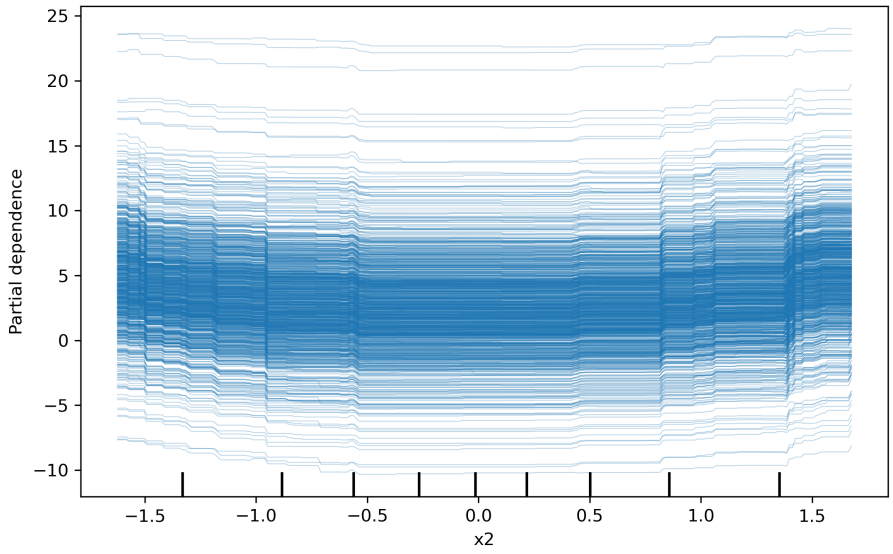

Figure 6: ICE Plot: x2

Figure 6 presents ICE plots for feature x_2. Each of the faint blue curves represents the prediction function for a single observation as x_2 varies, while the thick blue line shows the average partial dependence (equivalent to the PDP). The individual curves closely follow the average relationship, with relatively little heterogeneity across observations. This pattern

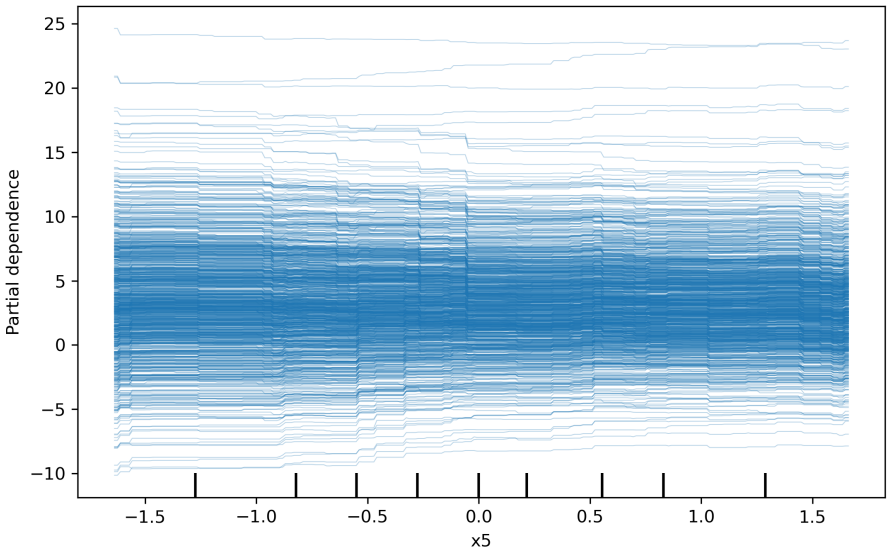

Figure 7: ICE Plot: x5

suggests that the quadratic effect of x_2 is fairly consistent across the dataset and that interaction effects involving x_2 are minimal. Indeed, we know from Equation (1) that interactions involving x_2 are not included in the data-generating process.

In contrast, Figure 7 tells a different story. The ICE curves for x_5 exhibit a degree of heterogeneity, with some observations showing positive relationships, others showing negative relationships, and others showing relatively flat relationships. The average partial dependence (blue line) is rather flat, but this masks considerable variation in how x_5 affects individual predictions.

This heterogeneity is entirely consistent with our data-generating process, where x_5 enters only through interaction terms: $x_4 \cdot x_5$ and $x_5 \cdot x_9$. The effect of x_5 on the prediction depends critically on the values of x_4 and x_9 for each observation.

This example illustrates a crucial insight for applied economists: when features primarily enter through interaction effects, their marginal effects can vary dramatically across observations, and partial dependence plots may provide misleading impressions of their overall importance. ICE plots help identify these situations by revealing the underlying heterogeneity that is often hidden by averaging. This reinforces the importance of considering interaction terms explicitly in machine learning interpretation, as average effects may fail to capture important variation across observations.

The SHAP-based approaches discussed in the next subsection provide another lens for examining these feature relationships and, importantly, allow us to construct confidence intervals around our estimates.

4.2 Average SHAP Values with Confidence Interval

The process for creating binned SHAP values with confidence intervals follows a bootstrapping procedure that should be familiar to economists who have experience with bootstrapping coefficient standard errors.[13] The algorithm is as follows:

1. Draw (with replacement) observations from the full sample
2. For the bootstrapped sample, refit the XGBoost model, using the same tuned hyperparameters
3. Calculate SHAP values for all observations in the bootstrapped sample

   Note that each bootstrap sample involves retraining the model and then recalculating SHAP values; we are not resampling precomputed SHAP values, but re-estimating them from scratch using new data draws.
4. Calculate the average SHAP value within a specified number of bins (for a given feature)
5. Repeat steps 1–4 n times
6. Create an average and a confidence interval based on the range of values in Step 4, across the bootstrapped samples

This approach mirrors the familiar econometric practice of repeatedly estimating a model on bootstrapped samples and storing the coefficient estimates from each iteration to construct confidence intervals. The key difference is that instead of storing coefficients, we store the average SHAP values within each bin across the feature's range.

To make the visualization manageable and interpretable, Step 4 uses binning: the feature range is divided into a fixed number of intervals, and SHAP values are averaged within each bin. This approach reduces noise while maintaining the ability to observe the functional relationship between the feature and the target variable. The confidence intervals then reflect the uncertainty around these bin-specific averages across our bootstrap iterations.

Figures 8, 9, and 10 demonstrate this approach for three different features. The solid line in each figure represents the average SHAP value within each bin, while the shaded region shows the 95% confidence interval around these estimates.

Figure 8 shows the results for x_2, one of our important features. The SHAP values exhibit the expected quadratic pattern. Importantly, the confidence intervals are relatively narrow and do not include zero for most bins, indicating that we can be confident about the statistical significance of x_2's relationship with the target variable.[14]

[13] Code to implement the bootstrapping steps is included in the Github replication repo.

[14] Negative SHAP values when x_2 is near zero do not indicate a negative relationship, but rather that these feature values contribute to below-average predictions relative to the baseline, consistent with the quadratic relationship where x_2^2 is minimized at zero.

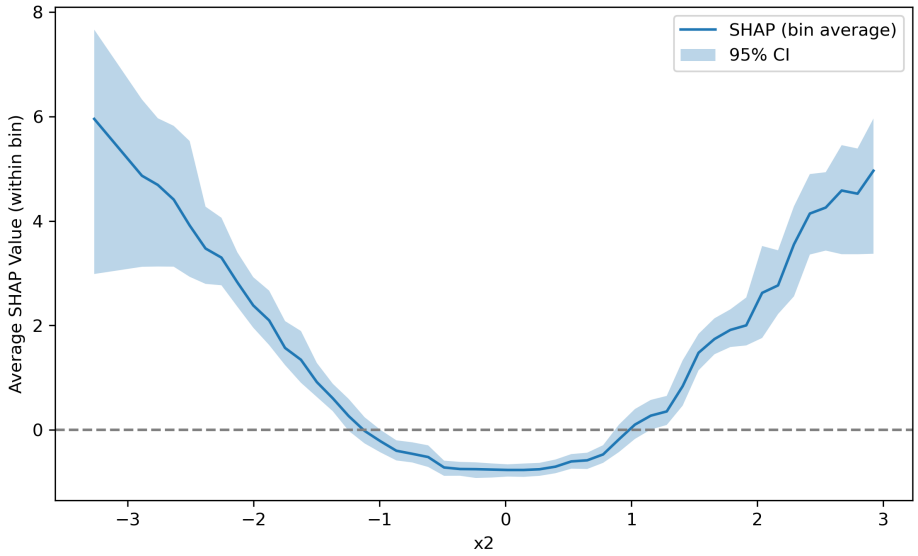

Figure 8: Average SHAP Values for x2 with 95% CI

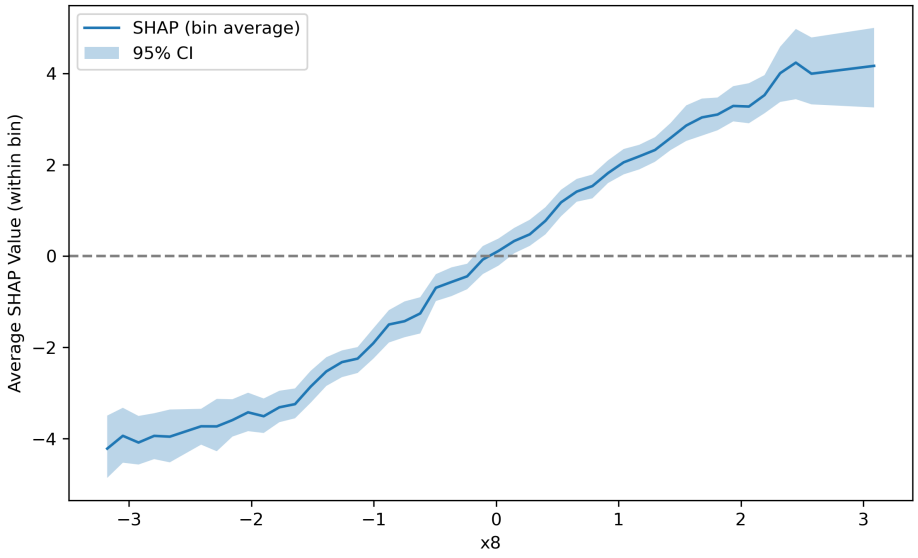

Figure 9: Average SHAP Values for x8 with 95% CI

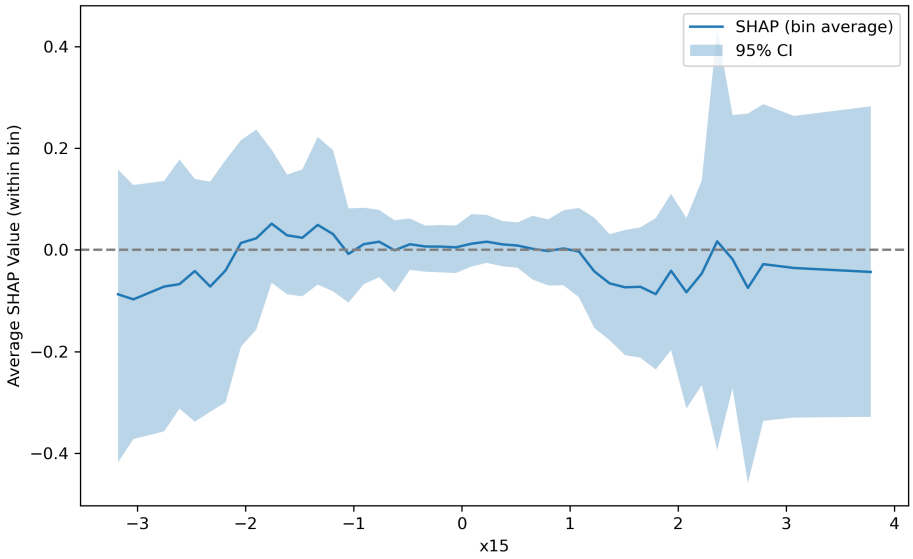

Figure 10: Average SHAP Values for x15 with 95% CI

Figure 9 presents similar results for x_8, showing a clear linear relationship between the feature values and the SHAP contributions. The confidence intervals again exclude zero across most of the feature's range, consistent with our knowledge that x_8 has a strong linear effect in the true data-generating process.

Figure 10, which examines one of our noise variables, provides a particularly instructive example. Here, the average SHAP values (solid line) remain close to zero across the entire range of x_{15}, never exceeding 0.10 in absolute value. More importantly, the 95% confidence intervals consistently include zero for all bins. This pattern provides clear statistical evidence that x_{15} has no meaningful relationship with the target variable, which is exactly what we would expect to see when evaluating the "statistical significance" of a truly irrelevant variable. This visual equivalent of a non-significant coefficient in regression reinforces that x_{15} has no explanatory power in the model.

This approach, while maintaining the visual interpretation that we see from PDPs, incorporates additional statistical rigor. It provides formal statistical inference about feature importance using an approach that is well-known in econometrics (bootstrapping). In this way, it can help identify when apparent relationships in PDPs might not be statistically distinguishable from zero. This allows researchers to distinguish between genuine relationships and spurious patterns.

5 Conclusion

The goal of this paper is to demonstrate that machine learning models can be compatible with the rigorous statistical inference found in traditional econometric approaches. Machine

learning interpretation replaces coefficient tables and t-statistics with visual tools and bootstrapped confidence intervals, offering statistically robust insights into feature relationships.

The interpretation toolkit presented here offers economists several complementary approaches for understanding machine learning models. For identifying which variables matter most in predicting our target variable, we examined three methods: built-in feature importance (fast and model-specific), permutation importance (model-agnostic with uncertainty quantification), and SHAP-based importance (theoretically grounded and interaction-aware). In our controlled simulation example, all three approaches successfully identified the relevant features, demonstrating how these methods can recover complex nonlinear relationships without requiring the researcher to specify functional forms.

For understanding how variables influence predictions, partial dependence plots provide visualizations of average relationships, analogous to average marginal effects in nonlinear econometric models. ICE plots show observation-level predictions, revealing when these average relationships mask important heterogeneity, particularly in the presence of interaction effects. Finally, SHAP-based approaches with bootstrapped confidence intervals add the statistical rigor that economists expect, allowing researchers to distinguish between genuine relationships and spurious patterns.

Several practical guidelines emerge from this analysis. When the goal is quick model exploration, built-in feature importance and basic PDPs provide efficient starting points. Computation time for both of those tools, in contrast to some tools relying on bootstrapping, is brief, allowing for the quick exploration of basic relationships. For publication-level results, permutation importance with confidence intervals and SHAP-based visualizations offer the level of formal inference expected in academic research.

Machine learning interpretation tools excel where traditional econometric models face limitations. They can capture complex nonlinear relationships without requiring researchers to specify functional forms a priori, handle high-dimensional feature spaces efficiently, and reveal interaction effects that might be missed in linear specifications. At the same time, they preserve the interpretability and statistical standards that economists value.

The goal is not to replace traditional econometric reasoning but to enhance it. These tools are especially useful when relationships are expected to be nonlinear, when functional forms are unknown, or when the main objective is accurate prediction alongside meaningful interpretation. In such settings, properly interpreted machine learning models can provide insights that both complement and extend traditional economic analysis.

As machine learning continues to evolve, economists who master these interpretation techniques will be well-positioned to harness the predictive power of modern algorithms while maintaining the analytical rigor that defines strong empirical research. The framework presented in this paper offers a foundation for incorporating these tools into the applied economist's empirical toolkit.

References

Breiman, Leo (2001a), "Random forests." *Machine learning*, 45, 5–32.

Breiman, Leo (2001b), "Statistical modeling: The two cultures (with comments and a rejoinder by the author)." *Statistical Science*, 16, 199–231, URL http://dx.doi.org/10.1214/ss/1009213726.

Desai, Ajit (2023), "Machine learning for economics research: When what and how?" Working Paper, URL https://ssrn.com/abstract=4404772.

Garg, Prashant and Thiemo Fetzer (2024), "Causal claims in economics." CESifo Working Paper Series No. 11462, URL https://ssrn.com/abstract=5045487.

Gebru, Timnit, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, Erez Lieberman Aiden, and Li Fei-Fei (2017), "Using deep learning and google street view to estimate the demographic makeup of neighborhoods across the united states." *Proceedings of the National Academy of Sciences*, 114, 13108–13113, URL https://www.pnas.org/doi/abs/10.1073/pnas.1700035114.

Gentzkow, Matthew, Bryan Kelly, and Matt Taddy (2019a), "Text as data." *Journal of Economic Literature*, 57, 535–74, URL https://www.aeaweb.org/articles?id=10.1257/jel.20181020.

Gentzkow, Matthew, Jesse M. Shapiro, and Matt Taddy (2019b), "Measuring group differences in high-dimensional choices: Method and application to congressional speech." *Econometrica*, 87, 1307–1340, URL https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA16566.

Gladstone, Joe J., Sandra C. Matz, and Alain Lemaire (2019), "Can psychological traits be inferred from spending? evidence from transaction data." *Psychological Science*, 30, 1087–1096.

Goldstein, Alex, Adam Kapelner, Justin Bleich, and Emil Pitkin (2015), "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation." *Journal of Computational and Graphical Statistics*, 24, 44–65.

Hansen, Stephen, Michael McMahon, and Andrea Prat (2017), "Transparency and deliberation within the fomc: A computational linguistics approach." *The Quarterly Journal of Economics*, 133, 801–870, URL https://doi.org/10.1093/qje/qjx045.

Jean, Neal, Marshall Burke, Michael Xie, W. Matthew Alampay Davis, David B. Lobell, and Stefano Ermon (2016), "Combining satellite imagery and machine learning to predict poverty." *Science*, 353, 790–794, URL https://www.science.org/doi/abs/10.1126/science.aaf7894.

Knittel, Christopher R and Samuel Stolper (2025), "Using machine learning to target treatment: The case of household energy use." *The Economic Journal*, ueaf028, URL https://doi.org/10.1093/ej/ueaf028.

Owens, Mark F and Adam D Rennhoff (2018), "Motion picture production incentives and filming location decisions: a discrete choice approach." *Journal of Economic Geography*, 20, 679–709, URL https://doi.org/10.1093/jeg/lby054.

Owens, Mark F and Adam D Rennhoff (2025), "Government-owned broadband: Evidence from a causal forest." Working Paper.

Park, Dong Jin, Min Woo Park, Hyeram Lee, Yu-Jin Kim, Yejee Kim, and Young Ho Park (2021), "Development of machine learning model for diagnostic disease prediction based on laboratory tests." *Scientific Reports*, 11, 7567.

Wager, Stefan and Susan Athey (2018), "Estimation and inference of heterogeneous treatment effects using random forests." *Journal of the American Statistical Association*, 113, 1228–1242, URL https://doi.org/10.1080/01621459.2017.1319839.

A Appendix

A.1 Demonstration of SHAP Calculations

In this Appendix, I present a simplified example to illustrate the basic steps involved in computing SHAP values for individual observations.

Background: Suppose that we collect a sample to explain home prices as a function of the number of bedrooms and whether the house has a garage or not. To make the calculations easier to follow, suppose that our linear regression implies the following prediction for a home price:

$$\widehat{\text{Price}} = 100 + 50(\text{bedrooms}) + 20(\text{garage}) \tag{A.1}$$

Further suppose that all houses in the sample have either 1 or 2 bedrooms. The average number of bedrooms is 1.5 and the average (binary) garage value is 0.6.

This simplified linear example allows us to understand SHAP mechanics in a familiar setting before extending the concept to more complex models.

Goal: calculate the SHAP values for an observation with 2 bedrooms and 1 garage.

Step 1: Calculate the prediction for our observation. Using equation A.1, the predicted price for a house with 2 bedrooms and a garage is:

$$\widehat{\text{Price}} = 100 + 50(2) + 20(1) = \$220,000$$

Step 2: Calculate the baseline prediction. The baseline represents the prediction when all features are set to their sample averages:

$$\text{Baseline} = 100 + 50(1.5) + 20(0.6) = \$187,000$$

Step 3: Calculate SHAP value for bedrooms. The SHAP value for bedrooms measures its marginal contribution across all possible coalitions of other features. With only one other feature (garage), there are two coalitions to consider:

Coalition 1: When garage = 0

$$\begin{aligned} \text{With average bedrooms:} &\quad 100 + 50(1.5) + 20(0) = \$175,000 \\ \text{With observed bedrooms:} &\quad 100 + 50(2.0) + 20(0) = \$200,000 \\ \text{Marginal contribution:} &\quad \$200,000 - \$175,000 = \$25,000 \end{aligned}$$

Coalition 2: When garage = 1

$$\begin{aligned} \text{With average bedrooms:} &\quad 100 + 50(1.5) + 20(1) = \$195,000 \\ \text{With observed bedrooms:} &\quad 100 + 50(2.0) + 20(1) = \$220,000 \\ \text{Marginal contribution:} &\quad \$220,000 - \$195,000 = \$25,000 \end{aligned}$$

The SHAP value for bedrooms is the weighted average of these marginal contributions:

$$\text{SHAP}_{\text{bedrooms}} = \frac{1}{2} \times \$25,000 + \frac{1}{2} \times \$25,000 = \$25,000$$

Step 4: Calculate SHAP value for garage. Similarly, the SHAP value for garage considers its marginal contribution across coalitions of the bedrooms feature:

Coalition 1: When bedrooms = 1

$$\begin{aligned} \text{With average garage:} \quad & 100 + 50(1) + 20(0.6) = \$162,000 \\ \text{With observed garage:} \quad & 100 + 50(1) + 20(1.0) = \$170,000 \\ \text{Marginal contribution:} \quad & \$170,000 - \$162,000 = \$8,000 \end{aligned}$$

Coalition 2: When bedrooms = 2

$$\begin{aligned} \text{With average garage:} \quad & 100 + 50(2) + 20(0.6) = \$212,000 \\ \text{With observed garage:} \quad & 100 + 50(2) + 20(1.0) = \$220,000 \\ \text{Marginal contribution:} \quad & \$220,000 - \$212,000 = \$8,000 \end{aligned}$$

The SHAP value for garage is:

$$\text{SHAP}_{\text{garage}} = \frac{1}{2} \times \$8,000 + \frac{1}{2} \times \$8,000 = \$8,000$$

Step 5: Verify the results. The SHAP values should sum to the difference between the prediction and the baseline:

$$\text{Baseline} + \text{SHAP}_{\text{bedrooms}} + \text{SHAP}_{\text{garage}} = \$187,000 + \$25,000 + \$8,000 = \$220,000$$

This matches our original prediction from Step 1, confirming our calculations. The SHAP values show that having 2 bedrooms (versus the average of 1.5) contributes \$25,000 to the price, while having a garage (versus the average probability of 0.6) contributes \$8,000.

In this linear example, these values could be calculated directly from the coefficients, but it is worth understanding the basic concept, as the direction and magnitude of SHAP values are far less transparent when applied to complex models like random forests or XGBoost.

A Note on Numerical Computation of SHAP Values: In this simple example, each of our features has only two possible values: bedrooms can be either 1 or 2, and garage can be either 0 or 1. This dramatically limits the number of coalitions that we must consider when computing the SHAP value for one of our features. In real applications, we may have dozens of features (n). The number of coalitions grows exponentially with the number of features. Even with binary features, computing SHAP values for one feature requires evaluating 2^{n-1} coalitions, and with continuous or multi-valued categorical features, the computational burden becomes even more severe. The calculation of all features and/or all

observations leads to a massive computation exercise. This computational burden mirrors the challenge economists face when computing Shapley values in cooperative game theory for games with many players. Statistical packages, such as Python's `shap` library, typically use sampling-based approximation methods to estimate SHAP values without evaluating every possible coalition.

A.2 Supplemental SHAP Visualizations

Beeswarm Plot

Figure 11 displays a SHAP beeswarm plot, which summarizes the overall importance and direction of each feature's impact on the XGBoost model's predictions. Each dot represents an individual observation, with color indicating the feature value and horizontal position showing the SHAP value. Features are ordered by their average absolute impact across all observations, which corresponds to the SHAP-based feature importance shown earlier in Figure 4.

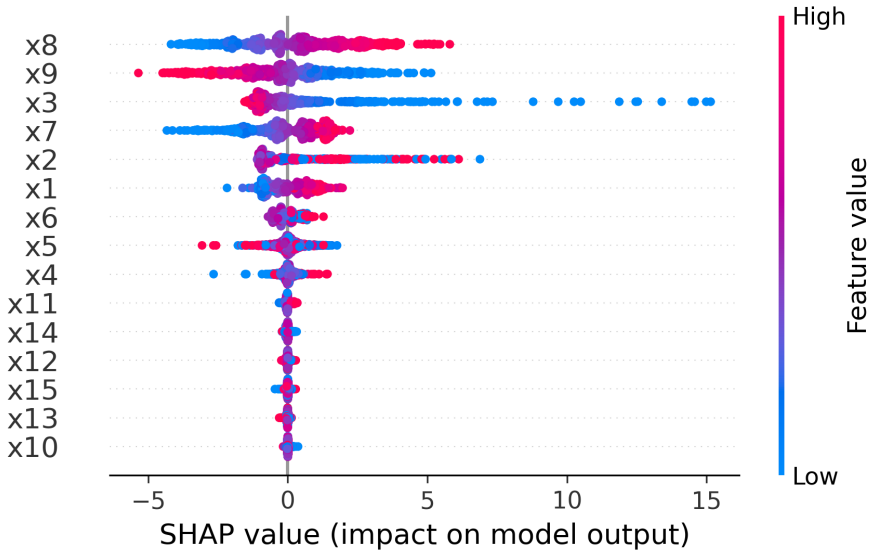

Figure 11: Beeswarm Plot for XGBoost Model

The feature values are color-coded: red indicates higher values of the feature, and blue indicates lower values. In these plots, the horizontal axis shows the SHAP value, which represents the contribution of a feature to the model's prediction for a given observation. Positive SHAP values increase the prediction, while negative values reduce it. For example, larger values of x_8 (red dots) are associated with positive SHAP values, while smaller values (blue dots) are associated with negative SHAP values.

Waterfall Plot

Figure 12 provides SHAP waterfall plots for Observations 1 and 2. These plots decompose each model prediction into the baseline (i.e., average prediction across the dataset) plus the cumulative contribution of each feature. The bars show how much each feature increases or decreases the prediction for a given observation. Features are ordered by impact (for a given observation), and the sign of each SHAP value determines whether the contribution pushes the prediction higher or lower. This illustrates how the same feature can have very different effects depending on the observation's context, reinforcing that SHAP values reflect individualized marginal contributions.

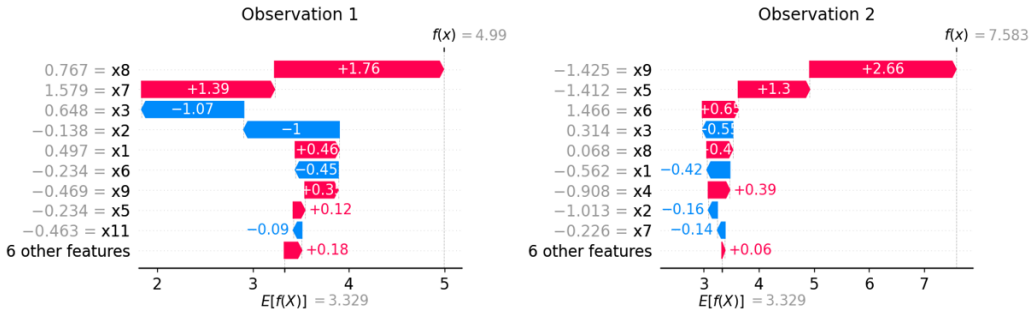

Figure 12: Waterfall Plots for Observations 1 & 2

For example, consider feature x_7: in Observation 1, $x_7 = 1.579$ and its SHAP value is $+1.39$, while in Observation 2, $x_7 = -0.226$ and its SHAP value is -0.14. This illustrates two key points. First, SHAP values are not fixed coefficients. Instead, they vary across observations depending on the context and interactions captured by the model. Second, they convey directional information: here, larger values of x_7 are associated with positive contributions to the prediction, consistent with the pattern seen in Figure 11.

Force Plot

Figure 13 presents SHAP force plots for Observations 1 and 2. Like the waterfall plots, these visualizations break down each prediction into the sum of feature contributions relative to a baseline prediction. The force plot offers a more compact summary of the same information: features that push the prediction higher are shown in red, while features that pull the prediction lower are shown in blue, with the final prediction indicated at the center.

Figure 13: Force Plots for Observations 1 & 2