Explainable AI in Machine Learning Models Using SHAP

Emily Hed

I. Introduction

~~As **artificial intelligence (AI)** advances, smart machines are increasingly integrated into our daily lives.~~ Recommendation systems, text analytics, autonomous vehicles, medical diagnostics, and other AI-driven technologies are shaping critical aspects of society, and many rely on these tools. However, as these systems become more complex, understanding a model's decision-making process is essential for detecting bias, debugging errors, ensuring transparency, and complying with regulations.

Explainable AI (XAI) elucidates AI decision-making using interpretability-focused machine learning techniques to understand model outputs. Current literature reveals two distinct XAI approaches: **machine learning (ML)** and **user experience (UX)**.

- The ML approach utilizes tools to explain model output. The field seeks to advance model performance, identify and mitigate bias, debug errors, and promote industry trust in AI systems [5].
- The UX approach investigates how users perceive and interact with AI explanations, drawing from psychology and **human-computer interaction (HCI)**. This approach evaluates whether a model's output is interpretable (users can understand it) and explainable (users can predict how changes in input affect output). This approach aims to foster user trust and reduce perceived risk [3].

XAI machine learning techniques can be categorized using model characteristics: global explanations, local explanations, model-specific, and model-agnostic (Figure 1).

SHAP (SHapley Additive exPlanations) is an XAI framework established on Shapley values from coalitional game theory. Shapley values provide a holistic view of how each feature (variable) influences a model's predictions [9]. SHAP's generic implementation is **model agnostic**, meaning SHAP can be applied to any model, regardless of underlying structure [1]. A key advantage of SHAP is its ability to provide both local and global model explanations [1].

- **Global explanations** describe the model as a whole, revealing which features exert the greatest influence on the model's predictions.
- **Local explanations** quantify how each feature impacts a single prediction.

SHAP is an open-source Python library that computes "SHAP values" [1] by considering all possible feature combinations and averaging their marginal contributions. SHAP values quantify the difference between the prediction and the model's global average prediction. This paper explores the mathematical properties of Shapley values, the underlying implementation of TreeSHAP, and tools in the SHAP library, including plots, explainers, and how Shapley values, despite their mathematical grounding, can be misleading for explainability.

[1]SHAP values are Shapley values applied to a machine learning model [8].

II. Overview

SHAP is an explainability method based on Shapley values in coalitional game theory. Coalitional game theory, also known as cooperative game theory, is a model that describes how groups of players, or coalitions, work together.

Let's look at an analogy:

> Imagine you're at a potluck dinner where each guest brings a dish. The overall meal enjoyment depends on the combination of dishes. Some dishes, like a well-seasoned main course, might have a greater impact on the meal's success, while others, like a simple side dish, contribute less. How do we determine how much each guest contributed to the overall meal enjoyment?

In this analogy, each guest represents a *feature* (variable), and their dish is the *contribution*. The overall meal satisfaction is the *prediction*. Shapley values determine how much each guest contributed to the meal enjoyment by evaluating different combinations of dishes and the resulting prediction. In other words, Shapley values quantify how each feature contributes to the overall prediction and how that prediction changes when joined to every possible combination of features [12].

Feature contribution [2] is a local measure of how a specific feature contributes to a single prediction whereas **feature importance** is a global measure that summarizes the overall impact of a feature across all predictions.

By considering all possible combinations of features, Shapley values provide a holistic view of how each feature influences the model's prediction.

III. Calculating Shapley Values

The Shapley value ϕ_i for a feature i is defined in Figure 2.

To better understand this formula, let's use an example. Consider three players, Player 1 (P1), Player 2 (P2), and Player 3 (P3) who enter a pie baking contest.

[2]*Feature contribution*, in the context of Shapley values, is synonymous with *Shapley values* and *SHAP values*. Therefore, the Shapley value or SHAP value *is* the feature contribution.

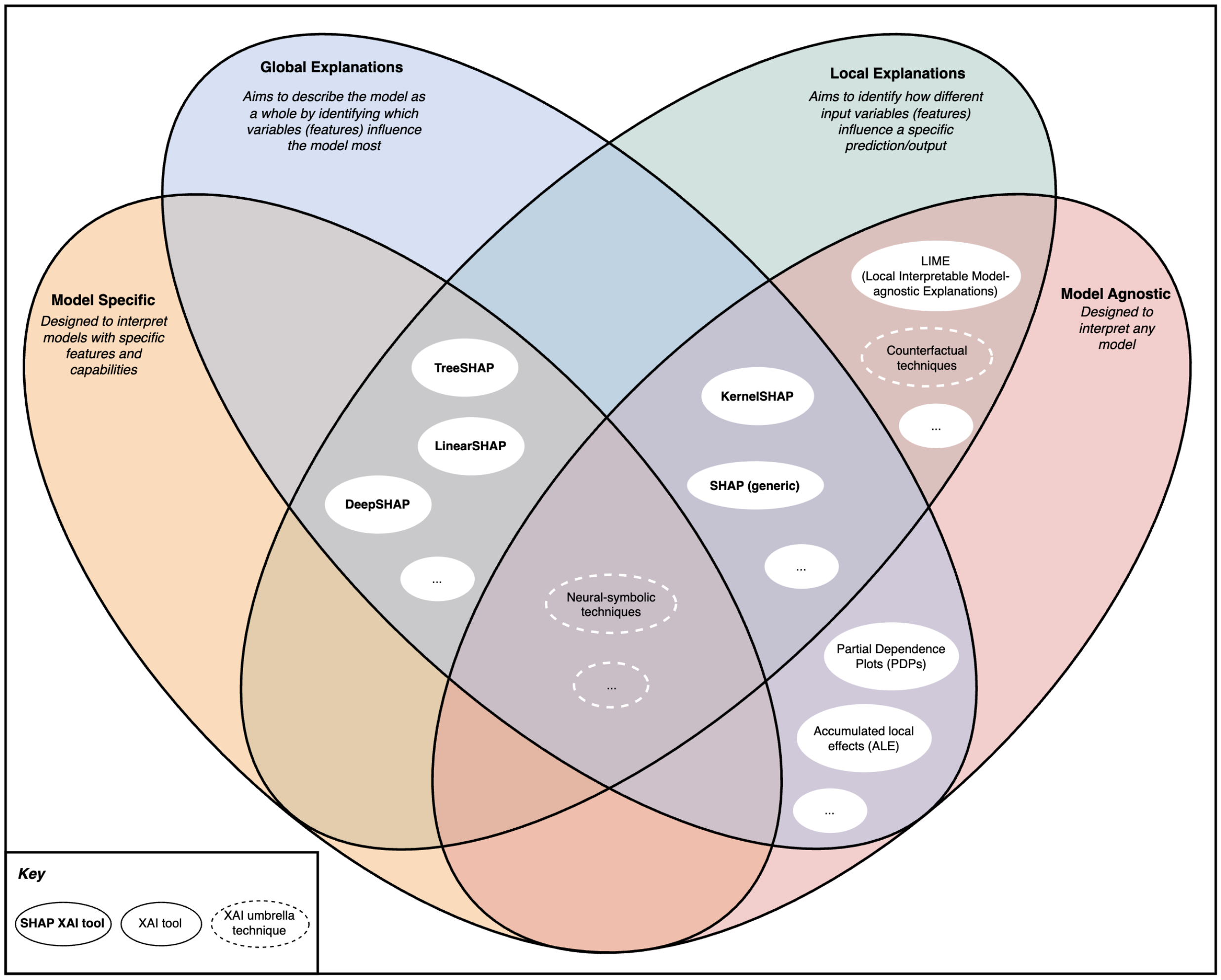
Global Explanations
Aims to describe the model as a whole by identifying which variables (features) influence the model most
Local Explanations
Aims to identify how different input variables (features) influence a specific prediction/output
Model Specific
Designed to interpret models with specific features and capabilities
Model Agnostic
Designed to interpret any model
LIME (Local Interpretable Model-agnostic Explanations)
Counterfactual techniques
...
TreeSHAP
LinearSHAP
DeepSHAP
...
KernelSHAP
SHAP (generic)
...
Neural-symbolic techniques
...
Partial Dependence Plots (PDPs)
Accumulated local effects (ALE)
...
Key
SHAP XAI tool
XAI tool
XAI umbrella technique

Fig. 1. A Venn diagram categorizing XAI techniques by global vs. local explanations and model-specific vs. model-agnostic approaches. An XAI technique can provide global and/or local explanations, but it cannot be both model-specific and model-agnostic. *TreeSHAP*, for example, provides both local and global explanations and is model-specific. Created by the author, compiled using resources from [5], [1].

$$\overbrace{\phi_i(v)}^{\text{Shapley value of } i} = \underbrace{\sum_{S \subseteq N \setminus \{i\}}}_{\text{Sum of all subsets without } i} \overbrace{\frac{|S|!(|N|-|S|-1)!}{|N|!}}^{\text{Weighted probability}} \underbrace{(v(S \cup \{i\}) - v(S))}_{\text{Marginal contribution of player } i \text{ to coalition } S}$$

Fig. 2. Shapley value equation [4].

P1, P2, and P3 decide to work together and place first, winning \$1,000 ($C_{123} = 1,000$). [3]

How should P1, P2, and P3 divide the prize money? P1 created the recipe, P2 measured and mixed the ingredients, and P3 baked the pie.

Assume we travel back in time (SHAP would re-query the model with different feature combinations) and test out different player combinations. Individually,

- P1 wins \$500 ($C_1 = \$500$),
- P2 wins \$500 ($C_2 = \$500$), and
- P3 wins \$0 ($C_3 = \$0$).

[3] C_{123} is the value of the coalition P1, P2, and P3.

If P1, P2, or P3 do not compete, none of these players win any prize money ($C_0 = 0$).

We also learn, when working as a team,

- P1 and P2 win \$750 ($C_{12} = \$750$),
- P1 and P3 win \$750 ($C_{13} = \$750$), and
- P2 and P3 win \$500 ($C_{23} = \$500$).

We can calculate the *marginal contributions* for each player by quantifying how much they increase the coalition's value when they join the coalition:

A. *P1's Marginal Contributions*

- P1 can join a coalition of P2 and P3:
  $C_{123} - C_{23} = \$1,000 - \$500 = \$500$

- P1 can join a coalition of P2:
  $C_{12} - C_2 = \$750 - \$500 = \$250$
- P1 can join a coalition of P3:
  $C_{13} - C_3 = \$750 - \$0 = \$750$
- P1 can join a coalition of no players:
  $C_1 - C_0 = \$500 - \$0 = \$500$

B. P1's Expected Marginal Contribution

To calculate the expected marginal contribution (Shapley value) of P1, we need to determine the probability that P1 makes these respective marginal contributions.

To calculate the probability of the first marginal contribution, $P(C_{123} - C_{23})$, we need to determine the likelihood that P1 makes a marginal contribution to a coalition of P2 and P3.

To start, we need to determine the number of ways a coalition of three players can form (we use three players, since this was the size of the team in the original scenario), assuming players join sequentially with equal chance:

- P1 + P2 + P3
- P1 + P3 + P2
- P2 + P3 + P1
- P2 + P1 + P3
- P3 + P1 + P2
- P3 + P2 + P1

P1 joins a coalition of P2 and P3 in 2 of the 6 scenarios, so $P(C_{123} - C_{23}) = {}^{2}\!/\!_{3} = {}^{1}\!/\!_{3}$.

We then multiply the weighted probability (${}^{1}\!/\!_{3}$) by the marginal contribution ($C_{123} - C_{23} = \$500$) and repeat this process for the remaining marginal contributions we get:

- $P(C_{123} - C_{23}) \cdot (C_{123} - C_{23}) = {}^{1}\!/\!_{3} \cdot \$500$
- $P(C_{12} - C_2) \cdot (C_{12} - C_2) = {}^{1}\!/\!_{6} \cdot \$250$
- $P(C_{13} - C_3) \cdot (C_{13} - C_3) = {}^{1}\!/\!_{6} \cdot \$750$
- $P(C_1 - C_0) \cdot (C_1 - C_0) = {}^{1}\!/\!_{3} \cdot \$500$

When we sum all values, we calculate P1's Shapley value = $500.

In other words, P1 should receive $500 of the original $1,000 prize money.

Following the same steps above, the expected marginal contributions of P2 and P3 are $375 and $125, respectively.

IV. Mathematical Properties and Strengths

SHAP values are considered a "definition of a fair weight" due to their mathematically desirable axioms [2], [6]:

- *Efficiency.* The sum of all feature contributions equals the difference between the prediction and the model's average, ensuring Shapley values are fairly distributed among features. For example, if the model predicts 60 and the average prediction is 50, the Shapley values sum to 10.
- *Symmetry.* If two features contribute equally to all possible coalitions, their Shapley values are equal.
- *Dummy.* A feature that does not impact the predicted value has a Shapley value of 0.
- *Linearity.* If two coalition games are combined, the Shapley value for each feature is the sum of its values in both games.

These properties ensure that SHAP is a fair and reliable method for explaining model predictions.

Another key advantage of SHAP is its versatility. It can explain any machine learning model, including enigmatic black-box models, and provides local and global explanations through plot visualization tools.

- *Local explanations* provide instance-specific explanations [1]. For example, consider a model that predicts the probability of a loan applicant defaulting based on factors like annual income and credit score. A local explanation lists the SHAP value for each feature, indicating how it influenced the final prediction. Figure 4 displays the global average prediction, $E[f(x)] = 0.28$, and shows that a *Monthly Debt* of 12316 *increased* the probability of this specific individual defaulting on their loan by 0.04.
- *Global explanations* average SHAP values across instances to reveal features that generally exert the greatest influence on predictions [1]. Figure 3 indicates *Current Loan Amount* typically has the greatest influence on individual predictions. The information gained from global analysis can be used to fine-tune the model and address potential biases [2].

V. Use Cases

SHAP can interpret model output for numeric and non-numeric data, such as text. The following examples illustrate SHAP's use cases based on a model's training data:

- *Tabular.* Structured data organized in rows and columns, typically stored in spreadsheets. Each row is an observation, and each column is a specific observation feature (e.g., age, gender, income). Examples include census data and medical records.
- *Text.* Unstructured natural language data such as documents, articles, social media posts, or emails.
- *Image.* Image data, represented as pixel grids with height, width, and color channels (e.g., RGB channels in a color image). Examples include satellite and medical images.
- *Genomic.* Genetic data from organisms, typically a sequence of nucleotides (A, T, C, G in DNA) or other biological features. Examples include DNA sequences and gene expression data.

SHAP is compatible with any ML model, regardless of training data.

VI. Explainers

SHAP is a model-agnostic and model-specific XAI tool, as the SHAP library possesses several "explainers," some of which are model-specific and others model-agnostic (Figure 1). For example, the model-specific `TreeExplainer` (known as *TreeSHAP*) is specifically designed for tree-based ML models (e.g., RandomForest, XGBoost, LightGBM, CatBoost). The model-agnostic `KernelExplainer` (known as *KernelSHAP*) is compatible with any ML model. Other explainers include LinearSHAP, DeepSHAP, and GradientSHAP, among others [8].

Algorithm 1 TreeSHAP [10]

```
procedure TS($x, tree = \{v, a, b, t, r, d\}$)
    $\phi$ = array of $len(x)$ zeros
    procedure RECURSE($j, m, p_z, p_o, p_i$)
        $m$ = EXTEND($m, p_z, p_o, p_i$)
        if $v_j \neq$ internal then
            for $i \leftarrow 2$ to $len(m)$ do
                $w$ = sum(UNWIND($m, i$)$.w$)
                $\phi_{m_i} = \phi_{m_i} + w(m_i.o - m_i.z)v_j$
            end for
        else
            $h, c = x_{d_j} \leq t_j \ ? \ (a_j, b_j) : (b_j, a_j)$
            $i_z = i_o = 1$
            $k$ = FINDFIRST($m.d, d_j$)
            if $k \neq$ nothing then
                $i_z, i_o = (m_k.z, m_k.o)$
                $m$ = UNWIND($m, k$)
            end if
            RECURSE($h, m, i_z r_h / r_j, i_o, d_j$)
            RECURSE($c, m, i_z r_c / r_j, 0, d_j$)
        end if
    end procedure
    procedure EXTEND($m, p_z, p_o, p_i$)
        $l = len(m)$
        $m$ = copy($m$)
        $m_{l+1}.(d, z, o, w) = (p_i, p_z, p_o, l = 0 \ ? \ 1 : 0)$
        for $i \leftarrow l - 1$ to 1 do
            $m_{i+1}.w = m_{i+1}.w + p_o m_i.w(i/l)$
            $m_i.w = p_z m_i.w[(l - i)/l]$
        end for
        return $m$
    end procedure
    procedure UNWIND($m, i$)
        $l = len(m)$
        $n = m_i.w$
        $m$ = copy($m_{1 \ldots l-1}$)
        for $j \leftarrow i - 1$ to 1 do
            if $m_i.o \neq 0$ then
                $t = m_j.w$
                $m_j.w = n \cdot l / (j \cdot m_i.o)$
                $n = t - m_j.w \cdot m_i.z((l - j)/l)$
            else
                $m_j.w = (m_j.w \cdot l)/(m_i.z(l - j))$
            end if
        end for
        for $j \leftarrow i$ to $l - 1$ do
            $m_j.(d, z, o) = m_{j+1}.(d, z, o)$
        end for
        return $m$
    end procedure
    RECURSE($1, [\ ], 1, 1, 0$)
    return $\phi$
end procedure
```

A. TreeSHAP

TreeSHAP is an explainer in the SHAP library specifically for tree-based models such as RandomForest, XGBoost, LightGBM, and CatBoost. Like all SHAP explainers, TreeSHAP determines the contribution of each input feature to the model's prediction. However, TreeSHAP is a unique explainer, as it precisely calculates SHAP values, whereas all other explainers approximate SHAP values through sampling methods [8].

Calculating SHAP values directly by testing all possible pathways through every tree is computationally infeasible for all but the smallest trees or datasets. However, the TreeSHAP algorithm, defined in Algorithm 1, allows us to compute the exact SHAP value in $O(TLD^2)$ time complexity and $O(D^2 + M)$ memory complexity [10], where

- T is the number of trees,
- L is the maximum number of leaves in any tree,
- M is the number of features, and
- D is the maximum depth of any tree.

Instead of explicitly listing and evaluating all possible 2^M feature subsets, TreeSHAP recursively tracks how the entire collection of possible subsets would distribute themselves down the specific prediction path, tracking the flow of subsets.

High-level summary. For example, imagine all possible feature combinations starting at the tree's root. As the algorithm traverses the path, dictated by the instance's values, TreeSHAP does not keep a list of which specific subsets go down each branch but maintains path summary information.

This summary information includes details about the proportions and aggregated weights of every possible subsets of features, up to the total number of features involved in the path so far. When the traversal ends at a leaf node, this information is used to calculate each feature's SHAP value.

Detailed walkthrough. Start at the root. The initial summary information represents the state before any feature splits, meaning all subsets are possible.

At each internal node along the path (including the root node)

1) Decide how to split using the feature and threshold information.
2) If the feature has not yet been encountered on the tree,
    a) Update summary information to reflect the consequences of this split by adjusting the proportion of subsets that include versus exclude the splitting feature, reflecting the number of instances now "following" this decision path.
    b) Update the aggregated weights associated with different subset sizes.
3) If the feature has been encountered,
    a) TreeSHAP performs a reversal procedure that temporarily undoes the effect of the previous split involving the same feature before updating the tracking information. This ensures that the update reflects the marginal impact of the current split decision relative to the state immediately preceding it. Without this reversal effect, TreeSHAP would measure the impact of the feature's second split using data that reflects the influence of its first split,

confounding its marginal contribution rather than isolating this lower-node decision.

b) After the reversal, update the summary information.

Every leaf node stores a prediction value, representing the model's output for that decision path. When the model reaches a leaf, the summary information reflects the cumulative effect of all path splits.

To determine the SHAP value for each feature encountered on the path, TreeSHAP uses the reversing procedure to remove each feature from the final summary state. The change observed in the expected output value during each removal reveals the feature's contribution (SHAP value).

For models composed of multiple trees, the process above is performed independently for each tree in the ensemble, using the same input features. A feature's SHAP value is calculated by adding its SHAP value from all trees.

VII. Visualization Methods

After selecting an appropriate explainer for a model's architecture, SHAP has several plots for visualizing the calculated SHAP values. Each tool provides unique data insights. Below are a few examples [8]:

- *Bar plot.* Displays each feature's *global average* SHAP value, representing its average contribution to the target variable (Figure 3). Features are ranked by influence, from most to least. Variations include local bar plot and cohort bar plot.
- *Waterfall plot.* Displays the vector SHAP value for each feature in a *local* prediction, illustrating each feature's contribution (Figure 4). The waterfall structure reveals the additive nature of positive and negative contributors from the model's base value (global average prediction) to the local prediction, building from the bottom up. The most important feature is listed first.
- *Heatmap.* Displays a plot with all instances on the x-axis, features on the y-axis, and SHAP values encoded on a color scale. Darker colors represent greater SHAP effects. Figure 5 illustrates the SHAP values for a model trained to predict whether individuals in the 1990s earned more than $\$50,000$ per year. The black bar chart to the right depicts each feature's global mean SHAP value, while the $f(x)$ line represents the predicted value for the specific instance.
- *Beeswarm plot.* Displays the distribution of SHAP values for each feature across all instances in the data set (Figure 6). Where SHAP values are dense, points are stacked vertically. The x-axis represents the SHAP value, indicating the importance of each feature in determining the prediction. The point's color represents the feature's value (red is high, blue is low). A red dot increases the prediction value, whereas a blue dot decreases.

VIII. Limitations

Shapley values in their original form suffer from exponentially increasing computational complexity as the number of features grows [1]. However, implementations of SHAP

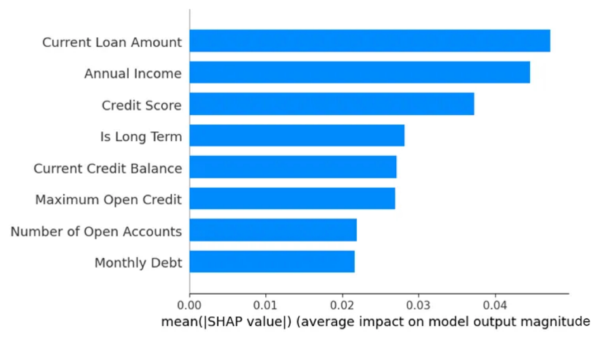

Fig. 3. A SHAP bar plot [11]. Displays each feature's *global average* SHAP value, representing its average contribution to the target variable. Features are ranked by influence, from most to least.

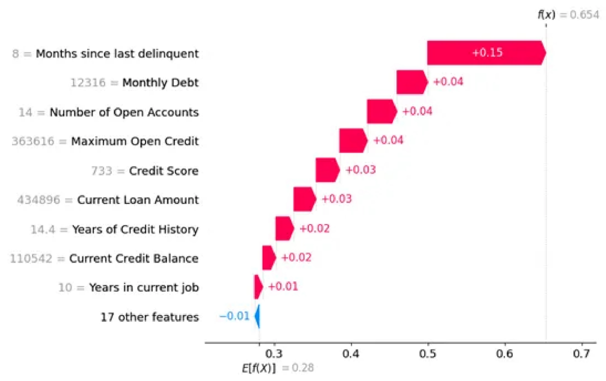

Fig. 4. A SHAP waterfall plot [11]. Displays the vector SHAP value for each feature in a *local* prediction, illustrating each feature's contribution. The waterfall structure reveals the additive nature of positive and negative contributors from the model's base value (global average prediction) to the local prediction, building from the bottom up. The most important feature is listed first.

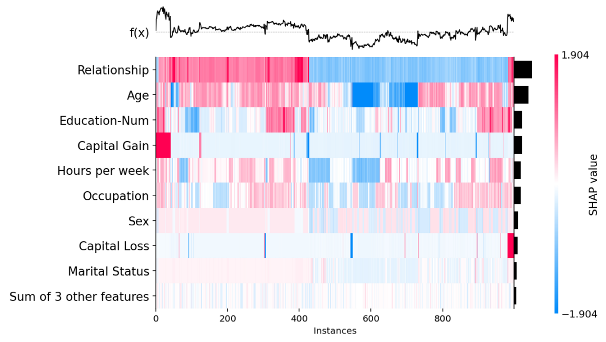

Fig. 5. A SHAP heatmap plot [8]. Displays a plot with all instances on the x-axis, features on the y-axis, and SHAP values encoded on a color scale. Darker colors represent greater SHAP effects. This chart depicts the SHAP values for a model trained to predict whether individuals in the 1990s earned more than $\$50,000$ per year. The black bar chart to the right depicts each feature's global mean SHAP value, while the $f(x)$ line represents the predicted value for the specific instance.

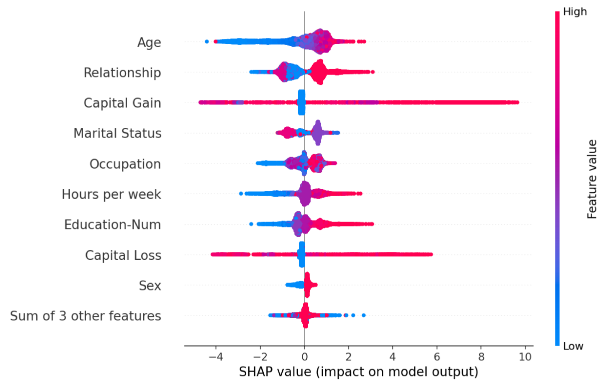

Fig. 6. A SHAP beeswarm plot [8]. Displays the distribution of SHAP values for each feature across all instances in the data set. Where SHAP values are dense, points are stacked vertically. The x-axis represents the SHAP value, indicating the importance of each feature in determining the prediction. The point's color represents the feature's value (red is high, blue is low). A red dot increases the prediction value, whereas a blue dot decreases.

estimate Shapley values using fewer computational resources using sampling techniques.

A 2023 paper, *The Inadequacy of Shapley Values for Explainability* [7], argues that Shapley values can provide misleading results even though Shapley values are theoretically grounded in game theory.

Using proofs and experimental data, the authors demonstrate that Shapley values can yield provably misleading information about the relative importance of features.

The paper identifies key issues with using Shapley values for feature importance:

- Irrelevant features can have a non-zero importance.
- Irrelevant features can be ranked higher than relevant ones.
- Relevant features can have zero importance.

How are Shapley values misleading despite satisfying mathematical fairness axioms? While the paper agrees that Shapley values satisfy the efficiency, symmetry, dummy, and linearity axioms, it argues that these axioms do not translate into a reliable measure of feature relevance. *Why?* Because there is a difference between *mathematical impact* (used by Shapley values) and *logical relevance*. The mathematical properties guaranteed by Shapley value axioms do not guarantee logical relevance [7].

A. Logical Relevance

Logical relevancy is defined using **Abductive Explanations (AXp)**. An AXp represents a minimal (irreducible) set of features whose values, if fixed according to the specific instance explained, are sufficient to guarantee the model's prediction outcome [7].

- A feature is logically relevant for a prediction if included in at least one AXp for that prediction.
- If a feature is not part of any AXp, it is logically irrelevant.

Let's use an analogy to clarify this concept: Imagine baking a Granny Smith apple pie. There might be several ways to bake this pie using the absolute minimum essential ingredients. One minimal recipe requires flour, butter, water, Granny Smith apples, sugar, and cinnamon. Another minimal recipe uses premade pie crust, Granny Smith apples, sugar, and cinnamon.

- An ingredient is "logically relevant" if it appears on at least one essential ingredient list. In this case, Granny Smith apples, sugar, cinnamon, flour, butter, water, and pre-made pie crust are all logically relevant because they're needed for at least one minimal recipe.
- An ingredient is "logically irrelevant" if it does not appear in any minimal recipe. For example, "a scoop of ice cream" or "confectioner's sugar" is not a minimum ingredient required to bake the pie.

Logical relevancy for the Granny Smith pie (the model's specific prediction) means identifying ingredients (features) that are absolutely necessary in at least one "bare-bones" recipe (the minimal sets, or AXps) to make that specific pie.

B. Mathematical Impact

Shapley values measure the average marginal contribution of a feature across all possible coalitions. Because the final Shapley value for a feature is an average of its contribution in many different coalition contexts, the specific contribution in one particular context can be eliminated through the averaging process. For example,

- A feature might be crucial in one minimal set, but if its contribution is zero or negative in many other coalitions, its Shapley value could be zero.
- Conversely, a feature might never be part of any minimal set (logically irrelevant), but has a small positive or negative contribution when added to various non-minimal coalitions. If these small contributions don't average to zero, the feature is incorrectly assigned a non-zero Shapley value.

These examples illustrate how the logical relevancy of a feature can diverge from the averaged contribution measured by its Shapley value for a specific instance.

C. SHAP Alternatives

The authors briefly introduce an alternative measure of feature importance: enumerate all the AXp's of an explanation problem, and rank the features by their occurrence in explanations, giving more weight to the smaller explanations. Such a measure ensures irrelevant features' score is 0. However, enumerating all AXp is generally exponentially complex, making it computationally infeasible for large problems [7].

D. Conclusion

In summary, Shapley values can be misleading because *mathematical impact* does not guarantee *logical relevance*. Despite satisfying fairness axioms, Shapley values may assign non-zero importance to irrelevant features or overlook relevant ones, as defined by AXp. These limitations demand caution if relying on Shapley values for model interpretability.

IX. Conclusion

As AI becomes more integrated into society, transparency in model decision-making is essential for debugging errors, detecting bias, ensuring transparency, and complying with regulations.

SHAP, a model-agnostic XAI tool, leverages Shapley values from cooperative game theory to provide a mathematical approach for determining feature contributions in machine learning models. SHAP provides both local and global explanations.

The SHAP Python library includes multiple visualization tools, such as bar plots, waterfall plots, heatmaps, and beeswarm plots, each offering unique insights into feature importance and model behavior.

SHAP is not without limitations. Because the mathematical impact measured by Shapley values doesn't always align with logical relevance, recent research argues that Shapley values, despite their mathematical grounding, can be misleading for explainability. While logical relevance alternatives exist, they can be computationally infeasible. Therefore, caution is needed when relying solely on Shapley values for model interpretability.

References

[1] Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *Artificial Intelligence*, 298:103502, 2021.

[2] Brain John Aboze. A comprehensive guide into shap values, May 2023.

[3] David A. Broniatowski. Psychological foundations of explainability and interpretability in artificial intelligence. Apr 2021.

[4] Hugh Chen, Scott Lundberg, and Su-In Lee. Understanding shapley value explanation algorithms for trees: Shapley values for trees, 2017. Accessed: 2025-04-04.

[5] Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, and Rajiv Ranjan. Explainable ai (xai): Core ideas, techniques, and solutions. 55(9), January 2023.

[6] Soufiane Fadel and Statistics Canada. Explainable machine learning, game theory, and shapley values: A technical review, Feb 2022.

[7] Xuanxiang Huang and Joao Marques-Silva. The inadequacy of shapley values for explainability, 2023.

[8] Scott Lundberg. Welcome to the shap documentation, 2018.

[9] Scott Lundberg. shap, Aug 2023.

[10] Scott Lundberg, Gabriel Erion, and Su-In Lee. *Consistent Individualized Feature Attribution for Tree Ensembles*. Mar 2019.

[11] Brendan Ng. Interpreting loan default prediction models using shap, Jan 2024.

[12] Marek Pawlicki, Aleksandra Pawlicka, Federica Uccello, Sebastian Szelest, Salvatore D'Antonio, Rafał Kozik, and Michał Choraś. Evaluating the necessity of the multiple metrics for assessing explainable ai: A critical examination. *Neurocomputing*, 602:128282, 2024.