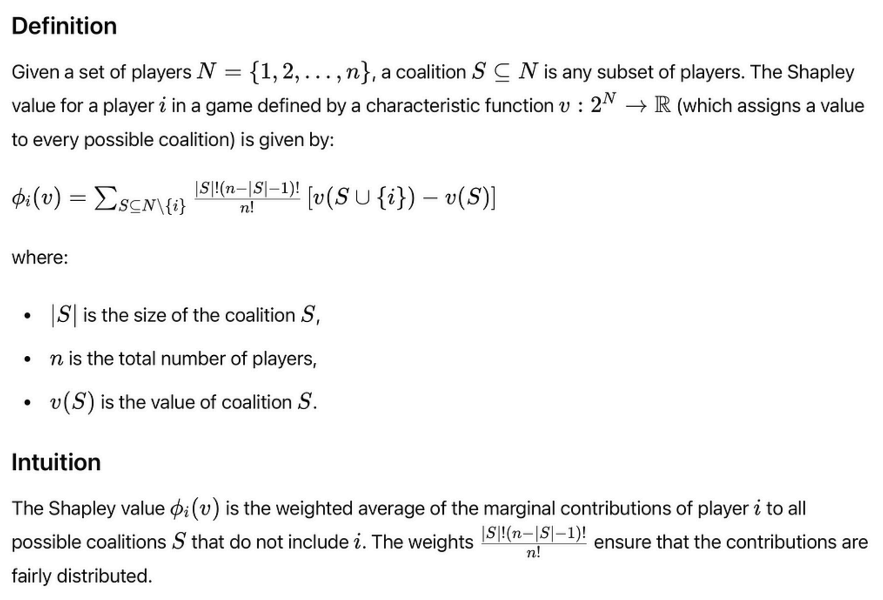
A Complete guide to SHAP Values

SHAP (SHapley Additive exPlanations) values are a method from game theory used to explain the output of machine learning models. In game theory, Shapley values are used to fairly distribute a total payoff among players based on their individual contributions. The core idea is to measure the contribution of each player to the total payoff, considering all possible combinations of players. For a given player, their Shapley value is the average marginal contribution they make across all possible coalitions of players.

Let's deep dive into the mathematics of Shapley Values in Cooperative Game Theory



Definition

Given a set of players $N = \{1, 2, \ldots, n\}$, a coalition $S \subseteq N$ is any subset of players. The Shapley value for a player i in a game defined by a characteristic function $v : 2^N \to \mathbb{R}$ (which assigns a value to every possible coalition) is given by:

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n-|S|-1)!}{n!} [v(S \cup \{i\}) - v(S)]$$

where:

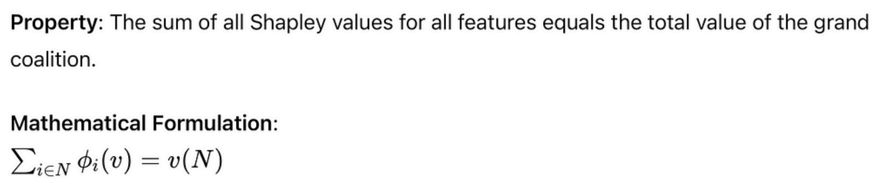
- $|S|$ is the size of the coalition S,
- n is the total number of players,
- $v(S)$ is the value of coalition S.

Intuition

The Shapley value $\phi_i(v)$ is the weighted average of the marginal contributions of player i to all possible coalitions S that do not include i. The weights $\frac{|S|!(n-|S|-1)!}{n!}$ ensure that the contributions are fairly distributed.

Now let's understand the properties of Shapley values helps in appreciating how they contribute to the total contribution of each player (or feature) in a fair and consistent manner. how each property contributes to the overall fairness and effectiveness of Shapley values:

1. Efficiency



Property: The sum of all Shapley values for all features equals the total value of the grand coalition.

Mathematical Formulation:
$$\sum_{i \in N} \phi_i(v) = v(N)$$

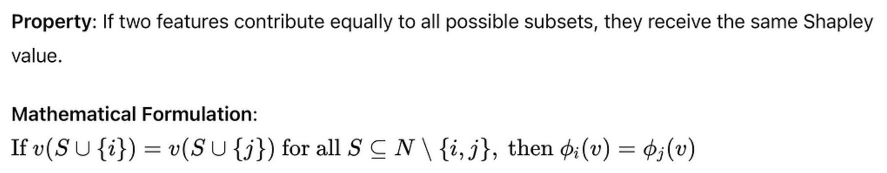
Explanation:

- **Grand Coalition:** In a cooperative game, the grand coalition is the set of all players working together. For a machine learning model, this represents the complete set of features used to make predictions.

- **Sum of Shapley Values:** This property ensures that the total value (or prediction) of the grand coalition is exactly accounted for by summing the individual Shapley values of all features.
- **Implication:** It guarantees that the Shapley values collectively add up to the model's prediction, ensuring that the total "payout" is distributed exactly among the features.

Example: In a house price prediction model, if the total predicted price is $500,000, the Shapley values of all features (like square footage, number of rooms, and location) will sum up to $500,000. This ensures that no value is unaccounted for, and every feature's contribution is properly recognized.

2. Symmetry


Property: If two features contribute equally to all possible subsets, they receive the same Shapley value.

Mathematical Formulation:
If $v(S \cup \{i\}) = v(S \cup \{j\})$ for all $S \subseteq N \setminus \{i, j\},$ then $\phi_i(v) = \phi_j(v)$

Explanation:

- **Equal Contribution:** This property states that if two features, say *i* and *j*, have identical contributions to any subset of features, then they should have the same Shapley value.
- **Fairness:** It ensures fairness by treating features equally if they are interchangeable in their contributions to the prediction. This is useful in scenarios where features are redundant or have the same effect on the outcome.

Example: If `square footage` and `number of rooms` both contribute equally to predicting house prices in every possible combination of features, they should receive the same Shapley value. This reflects their equal importance and ensures they are treated fairly in the model.

3. Dummy

Property: If a feature does not change the value of any coalition it joins, its Shapley value is zero.

Mathematical Formulation:
$\text{If } v(S \cup \{i\}) = v(S) \text{ for all } S \subseteq N, \text{ then } \phi_i(v) = 0$

Explanation:

- **No Contribution:** This property applies to a feature that has no effect on the model's prediction regardless of the presence or absence of other features. In other words, adding this feature to any subset of features does not change the prediction.
- **Zero Shapley Value**: If a feature does not influence the outcome in any scenario, its Shapley value is zero. This makes intuitive sense because it does not contribute to the model's prediction.

Example: In a model where `color` of the house has no impact on the price, its Shapley value will be zero. This simplifies the interpretation by confirming that `color` does not affect the prediction and should be disregarded in the analysis.

4. Additivity

Property: For two additive value functions (or models), the Shapley value of their sum is the sum of their individual Shapley values.

Mathematical Formulation:
$\phi_i(v + w) = \phi_i(v) + \phi_i(w)$

Explanation:

- **Additive Games**: When dealing with multiple games or models, if the total value function *v* is the sum of two value functions *v* and *w*, the Shapley values for the combined value function are simply the sum of the Shapley values for the individual value functions.
- **Model Combination**: This property is useful when combining multiple models or adding different contributions. It ensures that the Shapley values are consistent with the linear combination of value functions.

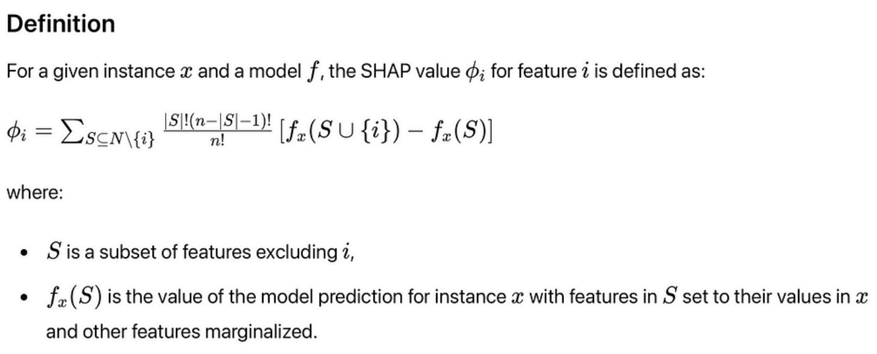
Example: If you have one model predicting house prices based on location and another based on number of bedrooms, the Shapley values for these models can be combined to get the Shapley values for a model that uses both features. This ensures that the total contribution of each feature is consistently accounted for.

How These Properties Work Together

- **Complete and Fair Distribution:** Efficiency ensures that the total contribution of all features equals the model's output, while symmetry guarantees fair treatment of equally contributing features.
- **Identification of Non-Contributors**: The dummy property identifies features that have no impact, simplifying the feature set and focusing on relevant features.
- **Combining Contributions:** Additivity supports combining contributions from multiple models or components, ensuring that the overall distribution remains fair and consistent.

SHAP Values in Machine Learning

In the context of machine learning, features are considered players, and the prediction function *f(x)* represents the characteristic function. The goal is to explain the prediction *f(x)* for a particular instance *x*.



Definition

For a given instance x and a model f, the SHAP value ϕ_i for feature i is defined as:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n-|S|-1)!}{n!} [f_x(S \cup \{i\}) - f_x(S)]$$

where:

- S is a subset of features excluding i,
- $f_x(S)$ is the value of the model prediction for instance x with features in S set to their values in x and other features marginalized.

Marginal Contribution

The term $f_x(S \cup \{i\}) - f_x(S)$ represents the marginal contribution of feature i to the prediction when added to the subset S.

Expected Value and Marginalization

To compute $f_x(S)$, we typically take the expected value of the model's output with respect to the missing features:

$$f_x(S) = \mathbb{E}[f(x)|x_S]$$

where x_S is the set of features in S.

Computational Complexity

Calculating SHAP values exactly is computationally expensive due to the need to evaluate the model on all possible subsets of features. For n features, this involves 2n evaluations. In practice, approximations and algorithms like Kernel SHAP and Tree SHAP are used to make computation feasible.

Kernel SHAP

Kernel SHAP is a model-agnostic approximation method that uses weighted linear regression to estimate SHAP values. It approximates the SHAP values by solving a least squares problem:

$$\arg\min_\phi \sum_{z' \subseteq x'} \left(f_x(z') - \phi_0 - \sum_{j=1}^{M} \phi_j z'_j \right)^2 \pi(z')$$

where:

- z' is a binary vector indicating the presence of features,
- $\pi(z')$ is a weighting kernel.

Tree SHAP

Tree SHAP is a method designed specifically for tree-based models (e.g., decision trees, random forests). It leverages the structure of trees to efficiently compute exact SHAP values in polynomial time.
Tree SHAP Algorithm

1. Traverse the tree to determine the contribution of each feature at each node.

2. Aggregate the contributions across all paths leading to a prediction.

3. Sum the contributions to obtain the SHAP value for each feature.

Efficiency

Tree SHAP exploits the hierarchical structure of trees to reduce the complexity from exponential to polynomial, making it feasible to compute exact SHAP values for large tree ensembles.

SHAP values provide a unified measure of feature importance by combining ideas from cooperative game theory with model interpretation. They offer a rigorous and theoretically grounded approach to explaining model predictions, ensuring fairness and consistency.

Limitations

While SHAP values provide a solid theoretical foundation and several practical advantages, they also have limitations. Here are some key limitations, explained with mathematical reasoning:

1. **Computational Complexity**:

- **Issue:** Calculating SHAP values can be computationally expensive, especially for models with many features.
- ***Explanation:*** *The exact computation of SHAP values involves considering all possible coalitions of features, which leads to an exponential number of combinations. Specifically, for a model with n features, the SHAP value for each feature requires evaluating* ***2^n*** *possible subsets of features.*
  The time complexity for exact SHAP value computation is ***O(2^n)****, which becomes infeasible for large* ***n****.*

2. **Independence Assumption**:

- **Issue:** Many implementations of SHAP, like Kernel SHAP, assume feature independence, which is often not true in real-world datasets.
- ***Explanation:*** *The assumption simplifies calculations but can lead to misleading interpretations when features are correlated.*

Let f be the model, ***S* ⊆ *N*** *be a subset of features, and* ***x_S*** *be the values of the features in* ***S****. SHAP values are computed under the assumption that* ***x_S*** *and* ***x_{−S}*** *(the values of the features not in* ***S****) are independent. This assumption simplifies the expectation* ***E[f(x)|x_S]*** *but is often unrealistic, leading to biased SHAP values.*

3. **Model-Specific SHAP Values:**

- **Issue:** The computation of SHAP values can vary depending on the model type, leading to inconsistencies.
- ***Explanation:*** *Different model types (e.g., tree-based models vs. linear models) may have different methods for approximating SHAP values, leading to different interpretations.*
  For tree-based models, SHAP values are computed using Tree SHAP, leveraging the structure of the tree for efficient computation. In contrast, for linear models, SHAP values are derived directly from model coefficients. This difference in computation methods can result in different SHAP value distributions.

4. **Interaction Effects:**

- **Issue:** SHAP values assume additive feature contributions, which can be misleading in the presence of strong interaction effects.
- ***Explanation****: The additive assumption means SHAP values do not capture complex interactions between features accurately.*
  If features ***x_i*** *and* ***x_j*** *interact strongly, the contribution of* ***x_i*** *might depend on the value of* ***x_j****. SHAP values approximate the contribution of* ***x_i*** *by averaging over all possible coalitions, potentially missing the interaction effect:*

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [f(S \cup \{i\}) - f(S)]$$

This equation assumes that $f(S \cup \{i\}) - f(S)$ is a good representation of x_i's contribution, which might not hold true for interacting features.

5. **Approximation Errors:**

- **Issue:** Many practical implementations use approximations to compute SHAP values to reduce computational cost, which can introduce errors.
- ***Explanation:*** *Approximations like sampling or simplifying assumptions can lead to inaccuracies in the computed SHAP values. Kernel SHAP, for example, uses a weighted linear regression approach to approximate SHAP values, introducing approximation error:*

$$\hat{\phi} = \arg\min_{\phi} \sum_{z \subseteq x} (\phi \cdot z - f(z))^2 \cdot \pi(z)$$

Here, $\pi(z)$ is a weighting term that can introduce bias if the number of samples is insufficient.

6. **Interpretability vs. Accuracy Trade-off:**

- **Issue:** The goal of interpretability can sometimes conflict with model accuracy or complexity.
- ***Explanation:*** *Simplifying models to make SHAP values more interpretable can lead to a loss of model performance.*
Reducing model complexity or enforcing linear relationships for better interpretability can increase the bias of the model, as per the bias-variance trade-off:
Total Error=Bias^2+Variance+Irreducible
Error Increasing interpretability often means increasing bias, reducing variance, and potentially increasing total error.

Conclusion

We have seen the mathematics behind SHAP values and how they are implemented in machine learning to determine feature importance (contribution of features to the model output) and model interpretability, as well as their limitations. In the next blog, we will explore the practical applications of SHAP values, demonstrating how they can be effectively used to interpret and explain model predictions, while also addressing the trade-offs and limitations encountered in real-world implementations.