Interpreting Machine Learning Models With SHAP

A Guide With Python Examples And Theory On Shapley Values

Christoph Molnar

Content

1 Preface

In my first book, “Interpretable Machine Learning,” I overlooked the inclusion of SHAP. I conducted a Twitter survey to determine the most frequently used methods for interpreting machine learning models. Options included LIME, permutation feature importance, partial dependence plots, and “Other.” SHAP was not an option.

To my surprise, the majority of respondents selected “Other,” with many comments highlighting the absence of SHAP. Although I was aware of SHAP at that time, I underestimated its popularity in machine learning explainability.

This popularity was a double-edged sword. My PhD research on interpretable machine learning was centered around partial dependence plots and permutation feature importance. On multiple occasions, when submitting a paper to a conference, we were advised to focus on SHAP or LIME instead. This advice was misguided because we should make progress for all interpretation methods, not just SHAP, but it underscores the popularity of SHAP.

SHAP has been subjected to its fair share of criticism: it’s costly to compute, challenging to interpret, and overhyped. I agree with some of these criticisms. In the realm of interpretable machine learning, there’s no perfect method; we must learn to work within constraints, which this book also addresses. However, SHAP excels in many areas: it can work with any model, it’s modular in building global interpretations, and it has a vast ecosystem of SHAP adaptations.

As you can see, my relationship with SHAP is a mix of admiration and frustration – perhaps a balanced standpoint for writing about SHAP. I don’t intend to overhype it, but I believe it’s a beneficial tool worth understanding.

2 Introduction

Machine learning models are powerful tools, but their lack of interpretability is a challenge. It's often unclear why a certain prediction was made, what the most important features were, and how the features influenced the predictions in general. Many people argue that as long as a machine learning model performs well, interpretability is unnecessary. However, there are many practical reasons why you need interpretability, ranging from debugging to building trust in your model.

> ℹ Interpretability
>
> I think of "interpretability" in the context of machine learning as a keyword. Under this keyword, you find a colorful variety of approaches that attempt to extract information about how the model makes predictions.

2.1 Interpreting to debug

Interpretability is valuable for model debugging, as illustrated by a study predicting pneumonia (Caruana et al. 2015). The authors trained a rule-based model, which learned that if a patient has asthma, they have a lower risk of pneumonia. Seriously? I'm no doctor, but that seems off. Asthma patients typically have a higher risk of lung-related diseases. It appears the model got it all wrong. However, it turns out that asthma patients in this dataset were less likely to get pneumonia. The indirect reason was that these patients received "aggressive care," such as early antibiotic treatment. Consequently, they were less likely to develop pneumonia. A typical case of "correlation does not imply causation" as you can see in Figure 2.1.

The problematic dependence on the asthma feature was only discovered due to the model's interpretability. Imagine if this scenario involved a neural network.

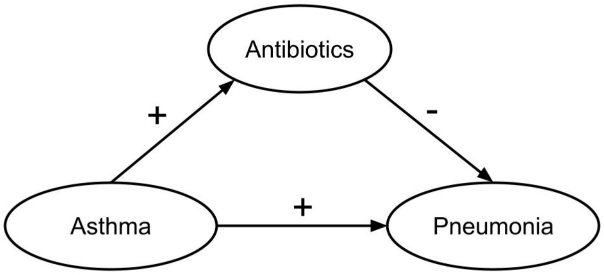

Figure 2.1: Asthma increases the likelihood of pneumonia. However, in the study, asthma also increased the (preemptive) use of antibiotics which generally protects against pneumonia and led to an overall lower pneumonia risk for asthma patients.

No rule would be apparent, stating “asthma ⇒ lower risk.” Instead, the network would learn this rule and conceal it, potentially causing harm if deployed in real-life situations.

Although you could theoretically spot the problem by closely examining the data and applying domain knowledge, it’s generally easier to identify such issues if you can understand what the model has learned. Machine learning models that aren’t interpretable create a distance between the data and the modeler, and interpretability methods help bridge this gap.

2.2 Users may create their own interpretations

Here’s a story about how the lack of interpretability led users of a model to develop their own incorrect interpretations. The story relates to sepsis, a life-threatening condition in which the body responds severely to an infection. As one of the most common causes of death in hospitals, sepsis is hard to diagnose, expensive to treat, and detrimental to patients, making early detection systems highly sought after.

Duke University and Duke Health Systems developed Sepsis Watch, an early warning system for sepsis in hospitals. This software system, based on deep neural networks, takes patient data as input and predicts whether the patient is likely to develop sepsis (Elish and Watkins 2020). If the model detects a potential sepsis case, it triggers an alert that initiates a new hospital protocol for diagnosis and treatment. This protocol involves a rapid response team (RRT) nurse who monitors the alarms and informs the doctors, who then treat the patient. Numerous aspects of the implementation warrant discussion, especially the social implications of the new workflow, such as the hospital hierarchy causing nurses to feel uncomfortable instructing doctors. There was also considerable repair work carried out by RRT nurses to adapt the new system to the hospital environment. Interestingly, the report noted that the deep learning system didn't provide explanations for warnings, leaving it unclear why a patient was predicted to develop sepsis. The software merely displayed the score, resulting in occasional discrepancies between the model score and the doctor's diagnosis. Doctors would consequently ask nurses what they were observing that the doctors were not. The patient didn't seem septic, so why were they viewed as high-risk? However, the nurse only had access to the scores and some patient data, leading to a disconnect. Feeling responsible for explaining the model outputs, RRT nurses collected context from patient charts to provide an explanation. One nurse assumed the model was keying in on specific words in the medical record, which wasn't the case. The model wasn't trained on text. Another nurse also formed incorrect assumptions about the influence of lab values on the sepsis score. While these misunderstandings didn't hinder tool usage, they underscore an intriguing issue with the lack of interpretability: users may devise their own interpretations when none are provided.

2.3 Building trust in your models

Anecdotally, I've heard data scientists express their avoidance of certain models, like neural networks or gradient boosting, due to their lack of interpretability. This decision isn't always left to the developer or data scientist, but could be influenced by their environment: the end user of the model, the middle manager who needs to understand the model's limitations and capabilities, or the senior data scientist who prefers interpretable models and sees no need to change. A lack of interpretability can discourage the use of models deemed uninterpretable. The fear of unexplained outcomes or the inability to use the model in its intended way can be overwhelming. For instance, the coefficients in a linear regression model could be used to inform other decisions, or a dashboard might display explanations alongside model scores to facilitate others' engagement with the predictions.

2.4 The limitations of inherently interpretable models

Is the solution to exclusively use "inherently" interpretable models? These may include:

- Linear regression and logistic regression
- Generalized additive models
- Decision rules & decision trees

Inherently interpretable typically means the model is constructed in a way that allows for easy understanding of its individual components. The prediction may be a weighted sum (linear model) or based on comprehensible rules. Some have even argued for the use of such models exclusively when the stakes are high (Rudin 2019).

However, there are two problems.

Problem 1: The definition of an interpretable model is ambiguous. One group may understand linear regression models, while another may not due to lack of experience. Even if you accept a linear regression model as interpretable, it can easily be made perplexing. For instance, by log-transforming the target, standardizing the features, adding interaction terms, using harder-to-interpret features, or adding thousands of features, an inherently interpretable model can become uninterpretable.

Problem 2: The models with the highest predictive performance are often not inherently interpretable. In machine learning, a metric is usually optimized. Boosted trees often emerge as the best choice in many scenarios (Grinsztajn et al. 2022). Most people wouldn't deem them interpretable, at least not in their original form. The same can be said for transformers, the standard for text (large language models, anyone?), and convolutional neural networks for image classification. Furthermore, ensembles of models often yield the best results and they are clearly less interpretable as they combine multiple models. Hence, restricting model selection to inherently interpretable models might lead to inferior performance. This inferior performance could directly result in fewer sales, increased churn, or more false negative sepsis predictions.

So, what's the solution?

2.5 Model-agnostic interpretation is the answer

Model-agnostic methods like explainable artificial intelligence (XAI) or interpretable machine learning (IML) provide solutions for interpreting any machine learning model[1]. Despite the vast differences between machine learning models, from k-nearest neighbors to deep neural networks and support vector machines, model-agnostic methods are always applicable as they don't need knowledge of the model's inner mechanics, such as coefficients.

Consider playing fighting games on a console, where you push inputs (the controller) and observe the outcomes (the character fights). This is similar to how model-agnostic interpretable machine learning operates. The model is treated like a box with inputs and outputs; you manipulate the inputs, observe how the outputs change, and draw conclusions. Most model-agnostic interpretation methods can be summarized by the SIPA framework (Scholbeck et al. 2020):

- **S**ampling data.
- **I**ntervention on the data.
- **P**rediction step.
- **A**ggregating the results.

Various methods operate under the SIPA framework (Molnar 2022), including:

- Partial dependence plots, which illustrate how altering one (or two) features changes the average prediction.
- Individual conditional expectation curves, which perform the same function for a single data point.
- Accumulated Local Effect Plots, an alternative to partial dependence plots.

[1] The terms "Explainable AI" and "interpretable machine learning" are used interchangeably in this book. Some people use XAI more for post-hoc explanations of predictions and interpretable ML for inherently interpretable models. However, when searching for a particular method, it's advisable to use both terms.

- Permutation Feature Importance, quantifying a feature's importance for accurate predictions.
- Local interpretable model-agnostic explanations (LIME), explaining predictions with local linear models (Ribeiro et al. 2016).

SHAP is another model-agnostic interpretation method that operates by sampling data, intervening on it, making predictions, and then aggregating the results.

> Tip
>
> Even if you use an interpretable model, this book can be of assistance. Methods like SHAP can be applied to any model, so even if you're using a decision tree, SHAP can provide additional interpretation.

2.6 SHAP: An explainable AI technique

SHAP (Lundberg and Lee 2017a) is a game-theory-inspired method created to explain predictions made by machine learning models. SHAP generates one value per input feature (also known as SHAP values) that indicates how the feature contributes to the prediction of the specified data point. In the example in Figure 2.2, the prediction model estimates the probability of a person earning more than $50k based on that person's socio-economic factors. Some factors positively affect the predicted probability, while others negatively impact it. Understanding this figure isn't crucial at this point; it's simply a goal to keep in mind as we dive into the theory behind SHAP in the following chapters.

SHAP has gained popularity and is applied in various fields to explain predictive models:

- Identifying COVID-19 mortality factors (Smith and Alvarez 2021).
- Predicting heat wave-related mortality (Kim and Kim 2022).
- Wastewater treatment plant management (Wang et al. 2022).
- Genome-wide association studies (Johnsen et al. 2021).
- Accident detection (Parsa et al. 2020).
- NO2 forecasting (García and Aznarte 2020).
- Molecular design (Rodríguez-Pérez and Bajorath 2020).
- Gold price forecasting (Jabeur et al. 2021).

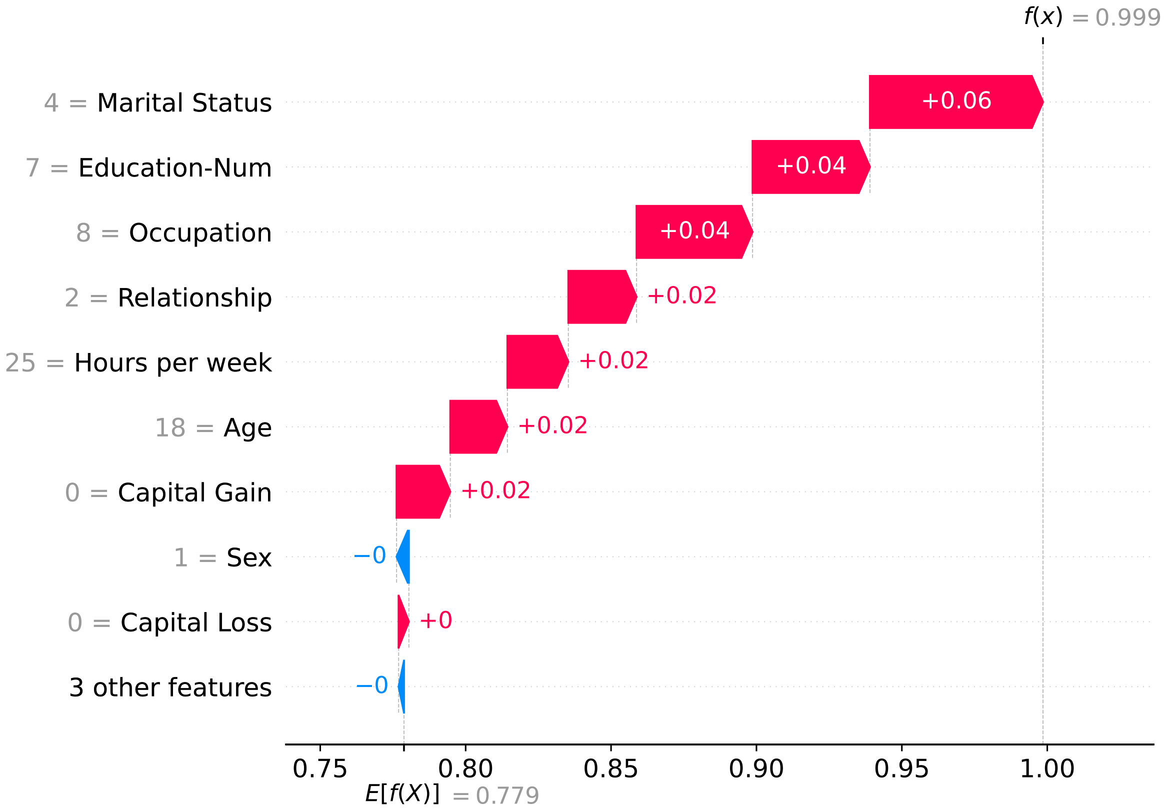

Figure 2.2: SHAP values to explain a prediction.

Given its wide range of applications, you are likely to find a use for SHAP in your work.

Before we talk about the practical application of SHAP, let's begin with its historical background, which provides context for the subsequent theory chapters.

3 A Short History of Shapley Values and SHAP

> By the end of this chapter, you will be able to:
>
> - Understand the key historical milestones of SHAP.
> - Explain the relationship between SHAP and Shapley values.

This chapter offers an overview of the history of SHAP and Shapley values, focusing on their chronological development. The history is divided into three parts, each highlighted by a milestone:

- 1953: The introduction of Shapley values in game theory.
- 2010: The initial steps toward applying Shapley values in machine learning.
- 2017: The advent of SHAP, a turning point in machine learning.

3.1 Lloyd Shapley's pursuit of fairness

Shapley values have greater importance than might initially be apparent from this book. These values are named after their creator, Lloyd Shapley, who first introduced them in 1953. In the 1950s, game theory saw an active period, during which many core concepts were formulated, including repeated games, the prisoner's dilemma, fictitious play, and, of course, Shapley values. Lloyd Shapley, a mathematician, was renowned in game theory, with fellow theorist Robert Aumann calling him the "greatest game theorist of all time"[1]. After World War II, Shapley completed his PhD at Princeton University with a thesis titled "Additive and Non-Additive Set Functions." In 1953, his paper "A Value for n-Person

[1] https://www.wsj.com/articles/lloyd-shapley-won-the-nobel-prize-for-economics-1923-2016-1458342678

Games" (Shapley et al. 1953) introduced Shapley values. In 2012, Lloyd Shapley and Alvin Roth were awarded the Nobel Prize in Economics[2] for their work in "market design" and "matching theory."

Shapley values serve as a solution in cooperative game theory, which deals with games where players cooperate to achieve a payout. They address the issue of a group of players participating in a collaborative game, where they work together to reach a certain payout. The payout of the game needs to be distributed among the players, who may have contributed differently. Shapley values provide a mathematical method of fairly dividing the payout among the players.

Shapley values have since become a cornerstone of coalitional game theory, with applications in various fields such as political science, economics, and computer science. They are frequently used to determine fair and efficient strategies for resource distribution within a group, including dividing profits among shareholders, allocating costs among collaborators, and assigning credit to contributors in a research project. However, Shapley values were not yet employed in machine learning, which was still in its early stages at the time.

3.2 Early days in machine learning

Fast forward to 2010. Shapley hadn't yet received his Nobel Prize in Economics, but the theory of Shapley values had been established for nearly 60 years. In contrast, machine learning had made tremendous strides during this period. In 2012, the ImageNet competition (Deng et al. 2009), led by Fei-Fei Li, was won for the first time by a team using a deep neural network (AlexNet) with a significant lead over the runner-up. Machine learning continued to advance and attract more research in many other areas.

While Shapley values had previously been defined for linear models, 2010 marks the beginning of model-agnostic estimation of Shapley values. In 2010, Erik Štrumbelj and Igor Kononenko published a paper titled "An efficient explanation of individual classifications using game theory" (Štrumbelj and Kononenko 2010), proposing the use of Shapley values to explain machine learning model predictions.

[2] It's not the real Nobel Prize, but the "Nobel Memorial Prize in Economic Sciences." Officially, it's termed the "Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel." This prize is a kind of surrogate Nobel award created by economists since they were not included in the original five Nobel Prizes.

In 2014, they further developed their methodology for computing Shapley values (Štrumbelj and Kononenko 2014).

However, this approach did not immediately gain popularity. Some possible reasons why Shapley values were not widely adopted at the time include:

- Explainable AI/Interpretable machine learning was not as widely recognized.
- The papers by Štrumbelj and Kononenko did not include code.
- The estimation method was still relatively slow and not suitable for image or text classification.

Next, we will look at the events that led to the rise of Shapley values in machine learning.

3.3 The SHAP Cambrian explosion

In 2016, Ribeiro et al. (2016) published a paper introducing Local Interpretable Model-Agnostic Explanations (LIME), a method that uses local linear regression models to explain predictions. This paper served as a catalyst for the field of explainable AI and interpretable machine learning. A more cautious claim might be that the paper's publication coincided with a growing interest in interpreting machine learning models. The prevailing sentiment at the time was a concern over the complexity of advanced machine learning algorithms, such as deep neural networks, and the lack of understanding of how these models generate their predictions.

Shortly after the LIME paper, in 2017, Scott Lundberg and Su-In Lee published a paper titled "A Unified Approach to Interpreting Model Predictions" (Lundberg and Lee 2017b). This paper introduced SHapley Additive exPlanations (SHAP), another method to explain machine learning predictions. The paper was published at NIPS, now known as NeurIPS[3]. NeurIPS is a major machine learning conference, and if your research is published there, it's more likely to draw attention. But what exactly did the SHAP paper introduce, given that Shapley values for machine learning were already defined in 2010/2014?

[3] The name NIPS faced criticism due to its association with "nipples" and its derogatory usage against Japanese individuals, leading to its change to NeurIPS.

Lundberg and Lee presented a new way to estimate SHAP values using a weighted linear regression with a kernel function to weight the data points[4]. The paper also demonstrated how their proposed estimation method could integrate other explanation techniques, such as DeepLIFT (Shrikumar et al. 2017), LIME (Ribeiro et al. 2016), and Layer-Wise Relevance Propagation (Bach et al. 2015).

Here's why I believe SHAP gained popularity:

- It was published in a reputable venue (NIPS/NeurIPS).
- It was a pioneering work in a rapidly growing field.
- Ongoing research by the original authors and others contributed to its development.
- The open-source `shap` Python package with a wide range of features and plotting capabilities

The availability of open-source code played a significant role, as it enabled people to integrate SHAP values into their projects.

ℹ Naming conventions

The naming can be slightly confusing for several reasons:

- Both the method and the resulting numbers can be referred to as Shapley values (and SHAP values).
- Lundberg and Lee (2017b) renamed Shapley values for machine learning as SHAP, an acronym for SHapley Additive exPlanations.

This book will adhere to these conventions:

- Shapley values: the original method from game theory.
- SHAP: the application of Shapley values for interpreting machine learning predictions.
- SHAP values: the resulting values from using SHAP for the features.
- `shap`: the library that implements SHAP.

"SHAP" is similar to a brand name used to describe a product category, like Post-it, Jacuzzi, Frisbee, or Band-Aid. I chose to use it since it's well-established in the community and it distinguishes between the general game-

[4] The `shap` package no longer uses Kernel SHAP by default, rendering the paper somewhat historical.

theoretic method of Shapley values and the specific machine learning application of SHAP.

Since its inception, SHAP's popularity has steadily increased. A significant milestone was reached in 2020 when Lundberg et al. (2020) proposed an efficient computation method specifically for SHAP, targeting tree-based models. This advancement was crucial because tree-boosting excels in many applications, enabling rapid estimation of SHAP values for state-of-the-art models. Another remarkable achievement by Lundberg involved extending SHAP beyond individual predictions. He stacked SHAP values, similar to assembling Legos, to create global model interpretations. This method was made possible by the fast computation designed for tree-based models. Thanks to numerous contributors, Lundberg continued to enhance the `shap` package, transforming it into a comprehensive library with a wide range of estimators and functionalities. Besides Lundberg's work, other researchers have also contributed to SHAP, proposing extensions. Moreover, SHAP has been implemented in other contexts, indicating that the `shap` package is not the only source of this method.

Given this historical context, we will begin with the theory of Shapley values and gradually progress to SHAP.

4 Theory of Shapley Values

By the end of this chapter, you will be able to:

- Understand the theory of Shapley values.
- Calculate Shapley values for simple games.
- Understand the axioms of Shapley values: efficiency, symmetry, dummy, and additivity.

To learn about SHAP, we first discuss the theory behind Shapley values from game theory. We will progressively define a fair payout[^fair] in a coalition of players and ultimately arrive at Shapley values (spoiler alert). [^fair]: There is no perfect definition of fairness everyone would agree upon. Shapley values define a very specific version of fairness, which can be seen as egalitarian.

4.1 Who's going to pay for that taxi?

Consider a concrete example that can be seen as a coalitional game: splitting the cost of a taxi ride. Alice, Bob, and Charlie have dinner together and share a taxi ride home. The total cost is $51. The question is, how should they divide the costs fairly?

View the taxi ride as a coalitional game: Alice, Bob, and Charlie form a coalition and receive a specific payout. In this case, the payout is negative (costs), but this doesn't change the fact that we can consider this as a coalitional game. To determine a fair distribution of the costs, we first pose simpler questions: How much would the ride cost for a random coalition of passengers? For instance, how much would Alice pay for a taxi ride if she were alone? How much would Alice and Bob pay if they shared a taxi? Let's suppose it would be $15 for Alice alone. Alice and Bob live together, but adding Bob to the ride increases the cost to $25, as he insists on a more spacious, luxurious taxi, adding a flat $10 to the ride costs. Adding Charlie to Alice and Bob's ride increases the cost to $51 since Charlie lives somewhat further away. We define the taxi ride costs for all possible combinations and compile the following table:

Passengers	Cost	Note
∅	$0	No taxi ride, no costs
{Alice}	$15	Standard fare to Alice's & Bob's place
{Bob}	$25	Bob always insists on luxury taxis
{Charlie}	$38	Charlie lives slightly further away
{Alice, Bob}	$25	Bob always gets his way
{Alice, Charlie}	$41	Drop off Alice first, then Charlie
{Bob, Charlie}	$51	Drop off luxurious Bob first, then Charlie
{Alice, Bob, Charlie}	$51	The full fare with all three of them

The coalition ∅ is a coalition without any players in it, i.e., an empty taxi. This table seems like a step in the right direction, giving us an initial idea of how much each person contributes to the cost of the ride.

4.2 Calculating marginal contributions for the taxi costs

We can take a step further by calculating the so-called marginal contributions of each passenger to each coalition. For example, how much additional cost does Alice incur when she joins a taxi with Bob already in it?

ℹ Marginal contribution

The marginal contribution of a player to a coalition is the value of the coalition *with* the player minus the value of the coalition *without* the player. In the taxi example, the value of a coalition is equal to the cost of the ride as detailed in the above table. Therefore, the marginal contribution of, for instance, Charlie to a taxi already containing Bob is the cost of the taxi with Bob and Charlie, minus the cost of the taxi with Bob alone.

Using the table, we can easily calculate the marginal contributions. Taking an example, if we compare the cost between the {Alice, Bob} coalition and Bob alone, we derive the marginal contribution of Alice, the "player", to the coalition {Bob}. In this scenario, it's \$25 - \$25 = \$0, as the taxi ride cost remains the same. If we calculate the marginal contribution of Bob to the {Alice} coalition, we get \$25 - \$15 = \$10, meaning adding Bob to a taxi ride with Alice increases the cost by \$10. We calculate all possible marginal contributions in this way:

Addition	To Coalition	Cost Before	Cost After	Marginal Contribution
Alice	∅	\$0	\$15	\$15
Alice	{Bob}	\$25	\$25	\$0
Alice	{Charlie}	\$38	\$41	\$3
Alice	{Bob, Charlie}	\$51	\$51	\$0
Bob	∅	\$0	\$25	\$25
Bob	{Alice}	\$15	\$25	\$10
Bob	{Charlie}	\$38	\$51	\$13
Bob	{Alice, Charlie}	\$41	\$51	\$10
Charlie	∅	\$0	\$38	\$38
Charlie	{Alice}	\$15	\$41	\$26
Charlie	{Bob}	\$25	\$51	\$26
Charlie	{Alice, Bob}	\$25	\$51	\$26

We're one step closer to calculating a fair share of ride costs. Could we just average these marginal contributions per passenger? We could, but that would assign equal weight to every marginal contribution. However, one could argue that we learn more about how much Alice should pay when we add her to an empty taxi compared to when we add her to a ride with Bob. But how much more informative?

One way to answer this question is by considering all possible permutations of Alice, Bob, and Charlie. There are $3! = 3 * 2 * 1 = 6$ possible permutations of passengers:

- Alice, Bob, Charlie
- Alice, Charlie, Bob

- Bob, Alice, Charlie
- Charlie, Alice, Bob
- Bob, Charlie, Alice
- Charlie, Bob, Alice

We can use these permutations to form coalitions, for example, for Alice. Each permutation then maps to a coalition: People who come before Alice in the order are in the coalition, people after are not. Since in a coalition the order of passengers doesn't matter, some coalitions will occur more often than others when we iterate through all permutations like this: In 2 out of 6 permutations, Alice is added to an empty taxi; In 1 out of 6, she is added to a taxi with Bob; In 1 out of 6, she is added to a taxi with Charlie; And in 2 out of 6, she is added to a taxi with both Bob and Charlie. We use these counts to weight each marginal contribution to continue our journey towards a fair cost sharing.

We could make different decisions regarding how to "fairly" allocate the costs to the passengers. For instance, we could weight the marginal contributions differently. We could divide the cost by 3. Alternatively, we could use solutions that depend on the order of passengers: Alice alone would pay \$15, when we add Bob it's +\$10, which would be his share, and Charlie would pay the remainder. However, all these different choices would lead us away from Shapley values.

4.3 Averaging marginal contributions

In two of these cases, Alice was added to an empty taxi, and in one case, she was added to a taxi with only Bob. By weighting the marginal contributions accordingly, we calculate the following weighted average marginal contribution for Alice, abbreviating Alice, Bob, and Charlie to A, B, and C:

$$\frac{1}{6}(\underbrace{2 \cdot \$15}_{\text{A to }\emptyset} + \underbrace{1 \cdot \$0}_{\text{A to B}} + \underbrace{1 \cdot \$3}_{\text{A to C}} + \underbrace{2 \cdot \$0}_{\text{A to B,C}}) = \$5.50$$

We multiply by $\frac{1}{6}$ because 6 is the sum of the weights (2 + 1 + 1 + 2). That's how much Alice should pay for the ride: \$5.50.

We can calculate the contribution for Bob the same way:

$$\frac{1}{6}(\underbrace{2 \cdot \$25}_{\text{B to }\emptyset} + \underbrace{1 \cdot \$10}_{\text{B to A}} + \underbrace{1 \cdot \$13}_{\text{B to C}} + \underbrace{2 \cdot \$10}_{\text{B to A,C}}) = \$15.50$$

And for Charlie:

$$\frac{1}{6}(\underbrace{2 \cdot \$38}_{\text{C to }\emptyset} + \underbrace{1 \cdot \$26}_{\text{C to A}} + \underbrace{1 \cdot \$26}_{\text{C to B}} + \underbrace{2 \cdot \$26}_{\text{C to A,B}}) = \$30.00$$

The individual contributions sum to the total cost: \$5.50 + \$15.50 + \$30.00 = \$51.00. Perfect! And that's it, this is how we compute Shapley values (Shapley et al. 1953).

Let's formalize the taxi example in terms of game theory and explore the Shapley value theory, which makes Shapley values a unique solution.

4.4 Calculating Shapley values

The upcoming sections will use several game theoretic terms. Even though we've already used most of them in the previous example, here's an overview for reference.

Term	Math Term	Taxi Example
Player	$1, \ldots, \|N\|$	Passenger, for example Alice
Coalition of All Players	N	{Alice, Bob, Charlie}
Coalition	S	Any combination of passengers, ranging from ∅ to {Alice, Bob, Charlie}.
Size of a Coalition	$\|S\|$	For example, $\|\{\text{Alice}\}\| = 1$, $\|\{\text{Alice}, \text{Bob}, \text{Charlie}\}\| = 3$
Value Function	$v()$	Defined by the table showing all possible arrangements of passengers in the taxi
Payout	v(N)	\$51, the cost of the taxi ride with all passengers

Term	Math Term	Taxi Example
Shapley Value	ϕ_j	For example, $\phi_1 = \$5.50$ for Alice, $\phi_2 = \$15.50$ for Bob, and $\phi_3 = \$30$ for Charlie.

The value function v can also be referred to as the characteristic function.

We have explored how to calculate Shapley values through the taxi ride example. Now, let's formalize Shapley values for the general case:

$$\phi_j = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!(N - |S| - 1)!}{N!} (v(S \cup \{j\}) - v(S)) \tag{4.1}$$

The value function $v : P(N) \mapsto \mathbb{R}$ maps from all possible coalitions of N players to a real number, which represents the payout for that coalition. The formula is quite complex, so let's break it down.

- $v(S \cup \{j\}) - v(S)$: This is the core of the equation. It represents the marginal contribution of player j to coalition S. If j is Alice and $S = \{Bob\}$, then this part expresses how much more expensive the ride becomes when Alice joins Bob.
- $\sum_{S \subseteq N \setminus \{j\}}$: The entire formula is a sum over all possible coalitions without j. If we calculate the Shapley value for Alice, we sum over the coalitions: $\emptyset$, $\{Bob\}$, $\{Charlie\}$, and $\{Bob, Charlie\}$.
- $\frac{|S|!(N-|S|-1)!}{N!}$: This term determines the weight of a marginal contribution. $\emptyset$ and $N \setminus \{j\}$ get the highest weights. The $|N|!$ in the denominator ensures that the sum of the weights equals 1.

The complex formula isn't so intimidating after all!

> **Shapley value formula summary**
>
> The Shapley value is the weighted average of a player's marginal contributions to all possible coalitions.

4.5 The axioms behind Shapley values

We now have the formula, but where did it come from? Lloyd Shapley derived it (Shapley et al. 1953), but it didn't just materialize out of thin air. He proposed axioms defining what a fair distribution could look like, and from these axioms, he derived the formula. Lloyd Shapley also proved that based on these axioms, the Shapley value formula yields a unique solution.

Let's discuss these axioms, namely **Efficiency**, **Symmetry**, **Dummy**, and **Additivity**. An axiom is a statement accepted as self-evidently true. Consider the axioms as defining fairness when it comes to payouts in team play.

4.5.1 Efficiency

The efficiency axiom states that the sum of the contributions must precisely add up to the payout. This makes a lot of sense. Consider Alice, Bob, and Charlie sharing a taxi ride and calculating their individual shares, but the contributions don't equal the total taxi fare. All three, including the taxi driver, would find this method useless. The efficiency axiom can be expressed formally as:

$$\sum_{j \in N} \phi_j = v(N)$$

4.5.2 Symmetry

The symmetry principle states that if two players are identical, they should receive equal contributions. Identical means that all their marginal contributions are the same. For instance, if Bob wouldn't need the luxury version of the taxi, his marginal contributions would be exactly the same as Alice's. The symmetry axiom says that in such situations, both should pay the same amount, which seems fair.

We can also express symmetry mathematically for two players j and k:

If $v(S \cup \{j\}) = v(S \cup \{k\})$ for all $S \subseteq N \backslash \{j, k\}$, then $\phi_j = \phi_k$.

4.5.3 Dummy or Null Player

The Shapley value for a player who doesn't contribute to any coalition is zero, which seems quite fair. Let's introduce Dora, Charlie's dog, and consider her an additional player. Assuming there's no extra cost for including Dora in any ride, all of Dora's marginal contributions would be \$0. The dummy axiom states that when all marginal contributions are zero, the Shapley value should also be zero. This rule seems reasonable, especially as Dora doesn't have any money.

To express this axiom formally:

If $v(S \cup \{j\}) = v(S)$ for all $S \subseteq N \backslash \{j\}$, then $\phi_j = 0$.

4.5.4 Additivity

In a game with two value functions v_1 and v_2, the Shapley values for the sum of the games can be expressed as the sum of the Shapley values:

$$\phi_{j,v_1+v_2} = \phi_{j,v_1} + \phi_{j,v_2}$$

Imagine Alice, Bob, and Charlie not only sharing a taxi but also going out for ice cream. Their goal is to fairly divide not just the taxi costs, but both the taxi and ice cream costs. The additivity axiom suggests that they could first calculate each person's fair share of the ice cream costs, then the taxi costs, and add them up per person.

These four[1] axioms ensure the uniqueness of the Shapley values, indicating there's only one solution presented in the Shapley formula, Equation 4.1. The proof of why this is the case won't be discussed in this book, as it would be too detailed. Instead, it's time to relate this approach to explaining machine learning predictions.

[1] A fifth axiom called *Linearity* or *Marginality* exists, but it can be derived from the other axioms, so it doesn't introduce any new requirements for fair payouts.

5 From Shapley Values to SHAP

By the end of this chapter, you will be able to:

- Describe a prediction as a coalitional game.
- Explain how SHAP values are defined.
- Interpret Shapley value axioms for machine learning predictions.

We have been learning about Shapley values from coalitional game theory. But how do these values connect to machine learning explanations? The connection might not seem apparent – it certainly didn't to me when I first learned about SHAP.

5.1 A machine learning example

Consider the following scenario: You have trained a machine learning model f to predict apartment prices. For a specific apartment $x^{(i)}$, the model predicts $f(x^{(i)}) = 300,000$. Your task is to explain this prediction. The apartment has an area of 50 m^2 (538 square feet), is located on the 2nd floor, has a nearby park, and cats are banned. These features are what the model used to make the prediction.

The average prediction for all apartments in the data is €310,000, which places the predicted price of this specific apartment slightly below average. How much did each feature value contribute to the prediction compared to the average prediction? In the apartment example, the feature values `park-nearby`, `cat-banned`, `area-50`, and `floor-2nd` collectively led to a prediction of €300,000. Our goal is to explain the difference between the actual prediction (€300,000) and the average prediction (€310,000), which is a difference of -€10,000. Here's an example of what an answer might look like: `park-nearby` contributed €30,000; `area-50`

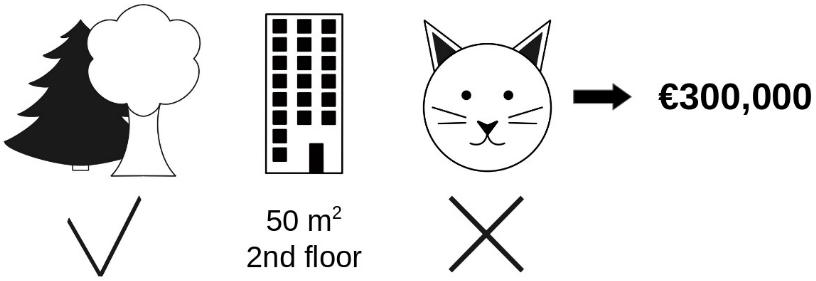

Figure 5.1: The predicted price for a 50 m^2 2nd floor apartment with a nearby park and cat ban is €300,000. Our goal is to explain how each of these feature values contributed to the prediction.

contributed €10,000; `floor-2nd` contributed €0; and `cat-banned` contributed -€50,000. The contributions add up to -€10,000, which is the final prediction minus the average predicted apartment price. From the Shapley theory chapter, we know that Shapley values can provide a fair attribution of a payout. We just need to translate concepts from game theory to machine learning prediction concepts.

5.2 Viewing a prediction as a coalitional game

A prediction can be viewed as a coalitional game by considering each feature value of an instance as a "player" in a game. The "payout" is the predicted value. We refer to this version of Shapley values adapted for machine learning predictions as "SHAP". Let's translate the terms from game theory to machine learning predictions one by one with the following table:

Concept	Machine Learning	Term
Player	Feature index	j
Coalition	Set of features	$S \subseteq \{1, \ldots, p\}$
Not in coalition	Features not in coalition S	$C : C = \{1, \ldots, p\} \backslash S$
Coalition size	Number of features in coalition S	$\|S\|$
Total number of players	Number of features	p

Concept	Machine Learning	Term
Total payout	Prediction for $x^{(i)}$ minus average prediction	$f(x^{(i)}) - \mathbb{E}(f(X))$
Value function	Prediction for feature values in coalition S minus expected	$v_{f,x^{(i)}}(S)$
SHAP value	Contribution of feature j towards payout	$\phi_j^{(i)}$

You may have questions about these terms, but we will discuss them shortly. The value function is central to SHAP, and we will discuss it in detail. This function is closely related to the simulation of absent features.

5.3 The SHAP value function

The SHAP value function, for a given model f and data instance $x^{(i)}$, is defined as:

$$v_{f,x^{(i)}}(S) = \int f(x_S^{(i)} \cup X_C) d\mathbb{P}_{X_C} - \mathbb{E}(f(X))$$

Note

The value function relies on a specific model f and a particular data point to be explained $x^{(i)}$, and maps a coalition S to its value. Although the correct notation is $v_{f,x^{(i)}}(S)$, I will occasionally use $v(S)$ for brevity. Another misuse of notation: I use the union operator for the feature vector: $x_S^{(i)} \cup X_C$ is a feature vector $\in \mathbb{R}^p$ where values at positions S have values from $x_S^{(i)}$ and the rest are random variables from X_C.

This function provides an answer for the simulation of absent features. The second part $\mathbb{E}(f(X))$ is straightforward: It ensures the value of an empty coalition $v(\emptyset)$ equals 0.

Confirm this for yourself:

$$\begin{aligned} v(\emptyset) = & \int f(X_1, \dots, X_p) d\mathbb{P}_X - E_X(f(X)) && (5.1)\\ = & E_X(f(X)) - E_X(f(X)) && (5.2)\\ = & 0 && (5.3) \end{aligned}$$

The first part of the value function, $\int f(x_S^{(i)} \cup X_C) dX_C$, is where the magic occurs. The model prediction function f, which is central to the value function, takes the feature vector $x^{(i)} \in \mathbb{R}^p$ as input and generates the prediction $\in \mathbb{R}$. However, we only know the features in set S, so we need to account for the features not in S, which we index with C.

SHAP's approach is to treat the unknown features as random variables and integrate over their distribution. This concept of integrating over the distribution of a random variable is called marginalization.

Marginalization

Integration of a function typically involves calculating the area under the curve. However, when integrating with respect to a distribution, certain portions under the curve are weighted more heavily based on their likelihood within the integral.

This means that we can input "known" features directly into the model f, while absent features are treated as random variables. In mathematical terms, I distinguish a random variable from an observed value by capitalizing it:

Feature value	Random variable
$x_j^{(i)}$	X_j
$x_S^{(i)}$	X_S
$x_C^{(i)}$	X_C
$x^{(i)}$	X

Let's revisit the apartment example:

Park	Cat	Area	Floor	Predicted Price
Nearby	Banned	50	2nd	€300,000

Informally, the value function for the coalition of park, floor would be:

$$v(\{\text{park}, \text{floor}\}) = \int f(x_{park}, X_{cat}, X_{area}, x_{floor}) d\mathbb{P}_{X_{cat,area}} - E_X(f(X)),$$

where $x_{park} = \text{nearby}$, $x_{floor} = 2$, and $E_X(f(X)) = 300.000$.

- The features 'park' and 'floor' are "present", so we input their corresponding values into f.
- The features 'cat' and 'area' are "absent", and thus are treated as random variables and integrated over.

5.4 Marginal contribution

We are gradually working our way up to the SHAP value. We've examined the value function, and the next step is to determine the marginal contribution. This is the contribution of feature j to a coalition of features S.

The marginal contribution of j to S is:

$$\begin{aligned} v(S \cup j) - v(S) &= \int f(x_{S\cup j}^{(i)} \cup X_{C\setminus j}) d\mathbb{P}_{X_{C\setminus j}} - \mathbb{E}(f(X)) \\ &\quad - \Big(\int f(x_S^{(i)} \cup X_C) d\mathbb{P}_{X_C} - \mathbb{E}(f(X)) \Big) \\ &= \int f(x_{S\cup j}^{(i)} \cup X_{C\setminus j}) d\mathbb{P}_{X_{C\setminus j}} \\ &\quad - \int f(x_S^{(i)} \cup X_C) d\mathbb{P}_{X_C} \end{aligned}$$

For instance, the contribution of 'cat' to a coalition of {park, floor} would be:

$$v(\{\text{cat}, \text{park}, \text{floor}\}) - v(\{\text{park}, \text{floor}\})$$

The resulting marginal contribution describes the change in the value of the coalition {park, floor} when the 'cat' feature is included. Another way to interpret the marginal contribution is that present features are known, absent feature values are unknown, so the marginal contribution illustrates how much the value changes from knowing j in addition to already knowing S.

5.5 Putting it all together

Combining all the terms into the Shapley value equation, we get the SHAP equation:

$$\phi_j^{(i)} = \sum_{S \subseteq \{1,\ldots,p\} \backslash j} \frac{|S|!\,(p-|S|-1)!}{p!} \cdot \left(\int f(x_{S\cup j}^{(i)} \cup X_{C\backslash j}) d\mathbb{P}_{X_{C\backslash j}} - \int f(x_S^{(i)} \cup X_C) d\mathbb{P}_{X_C} \right)$$

The SHAP value $\phi_j^{(i)}$ of a feature value is the average marginal contribution of a feature value $x_j^{(i)}$ to all possible coalitions of features. And that concludes it. This formula is similar to the one in the Shapley Theory Chapter, but the value function is adapted to explain a machine learning prediction. The formula, once again, is an average of marginal contributions, each contribution being weighted based on the size of the coalition.

5.6 Interpreting SHAP values through axioms

The axioms form the foundation for defining Shapley values. As SHAP values are Shapley values with a specific value function and game definition, they adhere to these axioms. This has been demonstrated by Štrumbelj and Kononenko (2010), Štrumbelj and Kononenko (2014), and Lundberg and Lee (2017b). Given that

SHAP follows the principles of Efficiency, Symmetry, Dummy, and Additivity, we can deduce how to interpret SHAP values or at least obtain a preliminary understanding. Let's explore each axiom individually and determine their implications for the interpretation of SHAP values.

5.6.1 Efficiency: SHAP values add up to the (centered) prediction

SHAP values must total to the difference between the prediction for $x^{(i)}$ and the expected prediction:

$$\sum_{j=1}^{p} \phi_j^{(i)} = f(x^{(i)}) - \mathbb{E}(f(X))$$

Implications: The efficiency axiom is prevalent in explainable AI and is adhered to by methods like LIME. This axiom guarantees that attributions are on the scale of the output, allowing us to interpret the results as contributions to the prediction. Gradients, another method for explaining model predictions, do not sum up to the prediction, hence in my opinion, are more challenging to interpret.

Symmetry: Feature order is irrelevant

If two feature values j and k contribute equally to all possible coalitions, their contributions should be equal.

Given

$$v_{f,x^{(i)}}(S \cup \{j\}) = v_{f,x^{(i)}}(S \cup \{k\})$$

for all

$$S \subseteq \{1, \dots, p\} \backslash \{j, k\}$$

then

$$\phi_j^{(i)} = \phi_k^{(i)}$$

Implications: The symmetry axiom implies that the attribution shouldn't depend on any ordering of the features. If two features contribute equally, they will receive the same SHAP value. Other methods, such as the breakdown method (Staniak and Biecek 2018) or counterfactual explanations, violate the symmetry axiom because two features can impact the prediction equally without receiving the same attribution. For example, the breakdown method also computes attributions, but does it by adding one feature at a time, so that the order by which features are added matters for the explanation. Symmetry is essential for accurately interpreting the order of SHAP values, for instance, when ranking features using SHAP importance (sum of absolute SHAP values per feature).

Dummy: Features not affecting the prediction receive SHAP values of 0

A feature j that does not alter the predicted value, regardless of the coalition of feature values it is added to, should have a SHAP value of 0.

Given

$$v_{f,x^{(i)}}(S \cup \{j\}) = v_{f,x^{(i)}}(S)$$

for all

$$S \subseteq \{1, \ldots, p\}$$

then

$$\phi_j^{(i)} = 0$$

Implications: The dummy axiom ensures that unused features by the model receive a zero attribution. This is an obvious implication. For instance, if a sparse linear regression model was trained, we can be sure that a feature with a $\beta_j = 0$ will have a SHAP value of zero for all data points.

5.6.2 Additivity: Additive predictions correspond to additive SHAP values

For a game with combined payouts $v_1 + v_2$, the respective SHAP values are:

$$\phi_j^{(i)}(v_1) + \phi_j^{(i)}(v_2)$$

Implications: Consider a scenario where you've trained a random forest, meaning the prediction is an average of numerous decision trees. The Additivity property ensures that you can compute a feature's SHAP value for each tree separately and average them to obtain the SHAP value for the random forest. For an additive ensemble of models, the final SHAP value equals the sum of the individual SHAP values.

> **i** Note
>
> An alternative formulation of the SHAP axioms exists where the Dummy and Additivity axioms are replaced with a Linearity axiom; however, both formulations eventually yield the SHAP values.

This chapter has provided theoretical SHAP values. However, we face a significant problem: In practice, we lack a closed-form expression for f and we are unaware of the distributions of X_C. This means we are unable to calculate the SHAP values, but, fortunately, we can estimate them.

6 Estimating SHAP Values

By the end of this chapter, you will be able to:

- Understand why SHAP values need to be estimated.
- Describe the permutation estimation method.
- Provide an overview of different SHAP estimation methods.

In the previous chapter, we applied the Shapley value concepts from game theory to machine learning. While exact Shapley values can be calculated for simple games, SHAP values must be estimated for two reasons:

- The value function utilized by SHAP requires integration over the feature distribution. However, since we only have data and lack knowledge of the distributions, we must use estimation techniques like Monte Carlo integration.
- Machine learning models often possess many features. As the number of coalitions increases exponentially with the number of features (2^p), it might become too time-consuming to compute the marginal contributions of a feature to all coalitions. Instead, we have to sample coalitions.

Let's assume we have a limited number of features for which we can still iterate through all coalitions. This allows us to focus on estimating the SHAP values from data without sampling coalitions.

6.1 Estimating SHAP values with Monte Carlo integration

Recap: SHAP values are computed as the average marginal contribution of a feature value across all possible coalitions. A coalition, in this context, is any subset of feature values, including the empty set and the set containing all feature values of the instance. When features are not part of a coalition, the prediction function still requires that we input some value. This problem was theoretically solved by integrating the prediction function over the absent features. Now, let's explore how we can estimate this integral using our apartment example.

The following figure evaluates the marginal contribution of the `cat-banned` feature value when added to a coalition of `park-nearby` and `area-50`. To compute the marginal contribution, we need two coalitions: {park-nearby, cat-banned, area-50} and {park-nearby, area-50}. For the absent features, we would have to integrate the prediction function over the distribution of floor, and floor + cat, respectively.

However, we don't have these distributions, so we resort to using Monte Carlo integration.

> **Monte Carlo integration**
>
> Monte Carlo integration is a method for approximating the integral of a function with respect to a random variable. It does this by drawing samples from the variable and averaging the function output for those samples. This technique allows us to sum over data instead of integrating over distributions.

Using Monte Carlo integration, we can estimate the value functions for our apartment by sampling the absent features from our data and averaging the predictions. In this case, the data are the other apartments. Sometimes, I'll refer to this data as background data.

> **Background data**
>
> The replacement of absent feature values with randomly drawn ones requires a dataset to draw from, known as the background data. This could be the same data that was used to train the model. The background data serves as the context for the interpretation of the resulting SHAP values.

Let's illustrate what sampling from the background data looks like by drawing just one sample for the Monte Carlo integration. Although a single sample results in a very unstable estimate of the integral, it helps us understand the concept.

Let's say the randomly sampled apartment has the following characteristics:

Park	Cat	Area	Floor	Predicted Price
Nearby	Allowed	100	1st	€504,000

Then, we replace the `floor-2nd` value of the original apartment with the randomly drawn `floor-1st` value. We then predict the price of the apartment with this combination (€310,000), which is the value function for the first coalition, v({park-nearby, cat-banned, area-50}).

Park	Cat	Area	Floor	Predicted Price
Nearby	Banned	50	1st	€310,000

Next, we replace `cat-banned` in the coalition with a random value of the cat allowed/banned feature from the same apartment that we sampled. In essence, we are estimating v({park-nearby, area-50}).

Park	Cat	Area	Floor	Predicted Price
Nearby	Allowed	50	1st	€320,000

In this scenario, the replaced value was `cat-allowed`, but it could have been `cat-banned` if we had drawn a different apartment. We predict the apartment price for the coalition of `park-nearby` and `area-50` to be €320,000. Therefore, the marginal contribution of `cat-banned` is €310,000 - €320,000 = -€10,000.

This estimate is based on the values of a single, randomly drawn apartment that served as a "donor" for the cat and floor feature values. This is not an optimal estimate of the marginal contribution as it relies on only one Monte Carlo sample. To obtain better estimates, we can repeat this sampling process and average the marginal contributions.

Now, let's get into the formalities. The value of a coalition of features S is estimated as:

$$\hat{v}(S) = \frac{1}{n} \sum_{k=1}^{n} \left(f(x_S^{(i)} \cup x_C^{(k)}) - f(x^{(k)}) \right)$$