EÖTVÖS LORÁND UNIVERSITY
Faculty of Science

Shapley Values and Explainable Artificial Intelligence

Barbara Babolcsay

Applied Mathematics MSc

Supervisor:
dr. András Zempléni

2025

Contents

Acknowledgements

I would like to express my sincere gratitude to my supervisor András Zempléni for his guidance throughout the development of this thesis. I am also thankful for his support during my MSc program and the professional experience I gained through working together.

I would like to thank my friends from both my bachelor's and master's studies for their continuous motivation and encouragement, which have consistently pushed me to achieve more.

Lastly, I am deeply grateful to my family and loved ones for their unconditional support and belief in me throughout my academic journey and beyond.

1. Introduction

In recent years, the field of artificial intelligence (AI) has seen rapid growth and widespread application across various domains, ranging from healthcare and finance to climate science and public safety. As models become increasingly powerful, they often function as ”black boxes”, providing accurate predictions without transparency into their decision-making processes. This opacity presents significant challenges related to trust, accountability, and interpretability.

To address these issues, the field of Explainable AI (XAI) has emerged, aiming to make complex models more understandable without sacrificing performance. Among the most popular approaches in XAI is SHapley Additive exPlanations (SHAP), which draws from cooperative game theory. Originally proposed by Lloyd Shapley in 1953, Shapley values offer a fair way to attribute the contribution of each feature to a model's prediction. This thesis investigates the application of Shapley values in machine learning, focusing on SHAP as a post-hoc explanation tool.

In Section 2, the theoretical foundations of Shapley values are introduced, tracing their origins in cooperative game theory. Section 3 provides a comprehensive description of XAI methods. Subsequently, Section 4 discusses the adaptation of Shapley values to machine learning models and their practical approximation, including the SHAP methodology. Section 5 presents the practical component of the thesis, beginning with numerical experiments on synthetic data, followed by a real-world case study focused on wildfire hazard estimation in the United Kingdom. Leveraging publicly available satellite and meteorological data from 2019 to 2023, classification models–such as XGBoost and neural networks–are trained and interpreted using SHAP values to identify the key features influencing fire hazard predictions.

This thesis connects game-theoretic fairness with practical interpretability in machine learning, demonstrating the value of Shapley-based explanations in applied AI.

2. Shapley Values in Game Theory

2.1. Game Theoretical Framework

One major field of game theory is cooperative games, where players can form coalitions in order to maximize their utility. In this section, I will discuss this topic in as much generality as necessary to establish a sound basis for the remainder of the thesis, based on [6] and [9].

In cooperative games, the utility can be transferable or non-transferable, depending on whether the profit can be redistributed among the players in the coalition or not. In the first case, with transferable utility, it is an important question, what share of the payoff each participant "deserves". Intuitively, this proportion should represent the players' contributions to the coalition's success.

We can define transferable utility (TU) games precisely with characteristic functions:

Definition 2.1. *Given n players, let* $v : 2^n \to \mathbb{R}$ *be a set-function that assigns a value to each subset* S *of the players as the maximum utility that coalition* S *can achieve. This function is the* ***characteristic function*** *and* $v(S)$ *is the* ***value*** *of* S*.*

The characteristic function should satisfy the following criteria:

- $v(\emptyset) = 0$,
- if $S \subset T \Rightarrow v(S) \leq v(T)$ (*monotonicity*).

After establishing the characteristic function of the game, we can also define the system of allocation among the players:

Definition 2.2. $\phi(v) \in \mathbb{R}^n$ *is an* ***allocation vector*** *to the characteristic function* v*, where* $\phi_i(v)$ *is the payoff share of player* i *in the game defined by* v*.*

We say that an allocation vector is in the **core** of the game if it is

- *efficient*: $\sum_{i=1}^{n} \phi_i = v([n])$, $[n] = \{1, \ldots, n\}$ and
- *stable*: $\sum_{i \in S} \phi_i \geq v(S)\ \forall S \subset [n]$,

meaning no sub-coalition has an incentive to break away from the grand coalition.[14]

Example 2.1 (Miners and Gold)**.** *Consider a group of miners who have discovered several large bars of gold. The total value of the loot to the group is determined by the number of bars they can carry home. It takes two miners to carry a single bar, so the value of the loot to any subset of k miners is given by:* $v(k) = \left\lfloor \frac{k}{2} \right\rfloor$.

If the total number of miners n is even, then the payoff vector $\phi = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right)$ *lies in the core. However, if the number of miners is odd, for instance,* $n = 3$*, then the core conditions require the following:*

$$\begin{aligned} \phi_1 + \phi_2 &\geq 1 \\ \phi_1 + \phi_3 &\geq 1 \\ \phi_2 + \phi_3 &\geq 1 \\ \phi_1 + \phi_2 + \phi_3 &= 1 \end{aligned}$$

This system of inequalities has no feasible solution, and therefore, ***the core is empty****.*

Example 2.2 (Splitting a Dollar)**.** *A parent offers their two children \$100 to split, but only if they can agree on how to divide it. If they fail to reach an agreement, each child receives only \$10.*

In cooperative game terms: $v(\{1,2\}) = 100, \quad v(\{1\}) = v(\{2\}) = 10$*. The core consists of all payoff vectors satisfying:*

$$\begin{aligned} \phi_1 &\geq 10 \\ \phi_2 &\geq 10 \\ \phi_1 + \phi_2 &= 100 \end{aligned}$$

This system clearly admits multiple solutions, meaning ***the core is non-empty and contains many possible agreements****.*

Remark 2.1. *In general, it is not always easy or feasible to decide whether a game has a non-empty core, but there are several special cases where we can be*

sure. For example, for simple games, where the value of a coalition is either 0 or 1, the core of the game is empty if and only if there are no veto players (a player that belongs to all winning coalitions).[14]

2.2. Shapley Values in n-person Games

Any allocation vector within the core can be considered a rational decision. However, as the examples above illustrate, the core may either be empty or contain multiple elements, which limits its practical applicability. To address this, it can be useful to define a unique allocation vector for each game, providing a more concrete and actionable solution concept while remaining fair.

L. S. Shapley introduced his axioms in his 1952 paper[13] that list the expected properties of the allocation vector of a game. Later, we will see that these axioms define a unique allocation vector for each game.

Shapley's axioms for the allocation vector ϕ are:

1. *Efficiency*: $\sum_{i=1}^{n} \phi_i = v([n])$,
2. *Symmetry*: if $v(S \cup \{i\}) = v(S \cup \{j\})\, \forall S$, if $i, j \notin S$ then $\phi_i = \phi_j$,
3. *Dummy*: if $v(S \cup \{i\}) = v(S)\, \forall S$, then $\phi_i = 0$ and
4. *Additivity:* $\phi_i(u + v) = \phi_i(u) + \phi_i(v)$ where u, v are characteristic functions.

The efficiency criterion is the same as in the definition of the core, meaning that the sum of players' shares should be equal to the value of the coalition of all players. The symmetry axiom guarantees the same payoff for players with the same added value, while the dummy axiom guarantees no share for players with no added value. The additivity property ensures that the allocation of different games do not affect each other.

Theorem 2.1 (Shapley's Theorem)**.** *Consider a fixed ordering of the players, defined by a permutation π of $[n]$. The players are arriving according to this permutation π, and define $\rho_i(v, \pi)$ to be the marginal contribution of player i at the time of his arrival. That is:*

$$\rho_i(v, \pi) = v(\pi\{1, \ldots, k\}) - v(\pi\{1, \ldots, k-1\}),$$

where $\pi(k) = i$. With this notation, Shapley's four axioms uniquely determine the functions ϕ_i, given by the so called ***random arrival formula****:*

$$\phi_i(v) = \frac{1}{n!} \sum_{\pi \in S_n} \rho_i(v, \pi).$$

The value assigned to the players by this allocation vector is the ***Shapley value****.*

Remark 2.2. *The Shapley value of i is the expected value $\rho_i(v, \pi)$, when π is chosen uniformly at random.*

Remark 2.3. *The Shapley value of player i can be written in the following form:*

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

Proof. [6] First let us show that the values given by the theorem's statement satisfy Shapley's axioms, then prove the uniqueness.

(A) Notice that for any fixed π, the dummy, efficiency, and additivity axioms would be satisfied with $\rho_i(v, \pi)$. Based on the theorem, we get the allocation vector by averaging $\rho_i(v, \pi)$ over all permutations, and since averaging preserves these properties, the final allocation vector $\phi_i(v)$ does as well.
To show that the symmetry property holds, define π^* to be the permutation obtained from π by swapping the positions of players i and j while leaving all other positions unchanged and $v(S \cup \{i\}) = v(S \cup \{j\})$ $\forall S$, if $i, j \notin S$. Then:

$$\rho_i(v, \pi) = \rho_j(v, \pi^*),$$

and since $\pi^{**} = \pi$,

$$\phi_i(v) = \frac{1}{n!} \sum_{\pi \in S_n} \rho_i(v, \pi) = \frac{1}{n!} \sum_{\pi \in S_n} \rho_j(v, \pi^*) = \frac{1}{n!} \sum_{\pi^* \in S_n} \rho_j(v, \pi^*) = \phi_j(v).$$

(B) For the uniqueness, let us introduce *S-veto games*, where S is a subset of players holding all the influence, so the value of the coalition T is 1, if $S \subseteq T$ and 0 otherwise. We can easily see that the axioms hold for the unique allocation vector, where $\phi_i(v) = 1/|S|$, if $i \in S$ and 0 otherwise. To extend the result to general games, we need the following lemma:

Lemma 2.1. *Every game can be expressed as a unique linear combination of S-veto games.*

Proof of lemma. For all nonempty $T \subseteq [n]$ we are looking for the solution of the following equation:

$$v(T) = \sum_{S \subseteq T, S \neq \emptyset} c_S v_S(T),$$

where v is the characteristic function of the general game and v_s is the S-veto game's. For all T, we get a linear equation system of $2n-1$ equations with the same amount of unknowns c_S. This has a unique solution, because the matrix of the system is upper triangular. □

The unique allocation vector that satisfies the axioms in the general case can be expressed as:

$$\phi_i(v) = \sum_{S \subseteq [n], S \neq \emptyset} c_S v_S = \sum_{S \subseteq [n], i \in S} c_S \phi_i(v_S) = \sum_{S \subseteq [n], i \in S} \frac{c_S}{|S|}.$$

With sections A and B, the proof of the theorem is complete. □

Remark 2.4. *The Shapley value has two other convenient properties:*

- *Determination by marginals: if the following holds for characteristic functions v_1 and v_2: $v_1(S + i) - v_1(S) = v_2(S + i) - v_2(S)\ \forall S \subseteq [n] \backslash i$, then $\phi_i(v_1) = \phi_i(v_2)$. A player's Shapley value is fully determined by how much they add to each coalition.*
- *Mutual profit: $\phi_i(v) - \phi_i(v \backslash j) = \phi_j(v) - \phi_j(v \backslash i)\ \forall i, j, v$. The mutual benefit between two players is symmetric.*

Example 2.3 (Four Stakeholders/Shapley-Shubik Index)**.** *Four individuals own shares in a corporation. Player i holds i units of stock, for* $i = 1, 2, 3, 4$*. A total of six shares is required to pass a resolution at the board meeting.*

The characteristic function $v(S)$ *is defined as follows:* $v(S) = 1$ *if the subset* $S \subseteq \{1, 2, 3, 4\}$ *collectively holds at least six shares, and* $v(S) = 0$ *otherwise.*

Notably, the following coalitions can pass a resolution:

$$v(\{2, 4\}) = v(\{3, 4\}) = v(\{1, 2, 3, 4\}) = 1$$

All other coalitions have a value of 0. In this setting, the Shapley value $\phi_i(v)$ *for each player i represents their voting power and is known as the* ***Shapley–Shubik power index****.*

By Lemma 2.1, we can write the value function of the game as the linear combination of S-veto games:

$$v = \sum_{S \subseteq N} c_S v_S$$

Solving this system yields:

$$v = v_{\{2,4\}} + v_{\{3,4\}} + v_{\{1,2,3\}} - v_{\{2,3,4\}} - v_{\{1,2,3,4\}}.$$

From this decomposition, we can compute the Shapley values:

$$\begin{aligned}
\phi_1(v) &= \frac{1}{3} - \frac{1}{4} = \frac{1}{12} \\
\phi_2(v) &= \frac{1}{2} + \frac{1}{3} - \frac{1}{3} - \frac{1}{4} = \frac{1}{4} \\
\phi_3(v) &= \frac{1}{4} \quad \textit{(by symmetry with player 2)} \\
\phi_4(v) &= 1 - (\phi_1 + \phi_2 + \phi_3) = 1 - \left(\frac{1}{12} + \frac{1}{4} + \frac{1}{4}\right) = \frac{5}{12}
\end{aligned}$$

It is interesting to observe that the players with two and three shares have equal power under this index, and also the core of this game is empty.

2.3. Shapley Values and Convex Games

The relation of the two main concepts discussed so far, the core and the Shapley values, is not clear in general. In fact, it is shown in [3], that it is NP-hard to decide if the Shapley value is in the core of a game or not. However, we can define a subset of cooperative games, where the inclusion is guaranteed.

Definition 2.3. *We call a game **convex**, if its characteristic function v is supermodular, meaning that $\forall X, Y \subseteq [n]$:*

$$v(X) + v(Y) \leq v(X \cup Y) + v(X \cap Y).$$

Theorem 2.2. *In convex games, the Shapley value is always in the core.*

Proof. [9] Let us consider a fixed ordering on the players, π, and define the payoff as before:

$$\rho_i(v, \pi) = v(\pi\{1, \ldots, k\}) - v(\pi\{1, \ldots, k-1\}),$$

where $\pi(k) = i$. We will show that this allocation vector lies in the core. Since the Shapley value is obtained as a convex combination of such vectors, and the core is a convex set, the Shapley value must also belong to the core.

For contradiction let us assume that $\exists U \subseteq [n] : \sum_{i \in U} \rho(i) < v(U)$, and choose a maximal such U. Let j be the smallest index such that $j \notin U$.

From the definition of the allocation vector we can derive:

$$v([j]) = v([j-1]) + \rho(j) = \ldots = \sum_{i \in [j]} \rho(i)$$
$$v([j-1]) = v([j-2]) + \rho(j-1) = \ldots = \sum_{i \in [j-1]} \rho(i).$$

From the supermodularity:

$$v([j]) + v(U) \leq v([j] \cap U) + v([j] \cup U) = v([j-1]) + v([j] \cup U)$$
$$\sum_{i \in [j]} \rho(i) + v(U) \leq \sum_{i \in [j-1]} \rho(i) + v([j] \cup U).$$

On the other hand,

$$\sum_{i \in [j]} \rho(i) + \sum_{i \in U} \rho(i) = \sum_{i \in [j-1]} \rho(i) + \sum_{i \in [j] \cup U} \rho(i),$$

and we assumed that $\sum_{i \in U} \rho(i) < v(U)$, so $\sum_{i \in [j] \cup U} \rho(i) < v([j] \cup U)$, which is in contradiction with U being maximal.

□

While the game-theoretic approach provides a mathematically grounded framework for evaluating fairness, contribution, and influence through concepts like the Shapley value, these ideas also resonate beyond cooperative game settings. In the context of artificial intelligence, especially as models grow more complex and opaque, understanding how individual features or inputs contribute to a model's decision becomes a central concern.

This naturally leads to the field of Explainable AI (XAI), which seeks to make the behavior of AI systems more transparent and interpretable. In the following section, we shift our focus to the broader landscape of XAI methods, exploring how different techniques aim to provide insights into model behavior at various stages and scopes.

3. AI Spring and XAI

In recent years, the development of artificial intelligence has accelerated significantly. With the rise of natural language processing (NLP) models and the availability of publicly accessible AI services, the field has become a topic of widespread public discourse and increasing attention.

As these models are being adopted in high-stakes domains such as healthcare, finance, and public safety, understanding their reliability has become critically important.

3.1. What is Explainable AI?

The need to explain the functioning of artificial intelligence (AI) methods is not new in scientific circles. However, as AI has become significantly more widespread, the demand for Explainable AI (XAI) has also grown rapidly.

But what exactly is the core issue? A commonly used term to describe many modern AI algorithms is the ”black-box model.” This refers to systems where the user receives little or no insight into the process that leads from the input to the output. Well-known and highly effective black-box models include ensemble methods such as XGBoost and Random Forest, kernel-based approaches, as well as neural network-based models like those used in natural language processing (NLP), including GPT (Generative Pretrained Transformers). This lack of transparency–referred to as opacity–can lead to significant challenges. When users cannot understand the model's behavior, it becomes difficult to identify potential biases or errors, or to hold the model accountable for its decisions. This opacity can result in a lack of trust and may hinder the deployment of AI in high-stakes domains.

In contrast, transparent or ”white-box” models offer insights into their internal mechanisms and the reasoning behind their predictions, making them easier to interpret and explain. Examples of interpretable models include classic machine learning algorithms such as logistic regression, decision trees, and k-nearest neighbors (k-NN). These models are often simpler in structure compared to black-box models, which typically leads to a trade-off between interpretability

and performance [12]. Nonetheless, recent advances in XAI have led to the development of state-of-the-art models that aim to bridge this gap, achieving both strong predictive performance and a degree of explainability.

3.2. Explanation Directions

The aim of XAI methodology is to bridge the gap between accurate black-box models and understandable white-box models by explaining the more complex models in different ways. I will summarize these directions based on the comprehensive reviews [10] and [5].

3.2.1. XAI Methods Categorized by Stage of Use

Minh et al. [10] categorize explainability methods according to the stage of the modeling process at which they are applied. I will summarize their classification in the following.

Pre-modeling Explainability

Pre-modeling explainability methods focus on the input data used for training the model, specifically targeting data preprocessing steps. These techniques are particularly valuable when dealing with very large datasets, which can be time-consuming to process directly.

Data analysis provides insights into the statistical properties of the dataset and helps identify potential issues, such as missing or corrupted data.

Data summarization techniques aim to generate a representative subset of the original large dataset, allowing the model to be trained efficiently on this smaller subset without significant loss of accuracy. Unsupervised methods, such as k-means clustering, are commonly employed for this purpose.

Data transformation involves enriching the dataset with useful metadata, which can include descriptions of data creation, preparation, collection methods, as well as relevant legal and ethical considerations. This additional information enhances the usability and interpretability of the data for downstream users.

Interpretable Models

The two main approaches in this category are the use of white-box models, which are *inherently interpretable*, and the combination of complex black-box models with interpretable models. The latter approach leads to *hybrid models*, where the simpler interpretable model aids in understanding the complex one by effectively reducing its complexity, resulting in a model that is both efficient and explainable.

Post-modeling Explainability

In order to preserve the accuracy of black-box models, it is often more effective to explain their decisions after they have been made–after the model is trained.

One category of these so-called post-modeling methods can be applied to a wide range of machine learning models (*model-agnostic methods*), since they treat the model as a black box. These methods rely on perturbing inputs and observing corresponding changes in outputs. Alternatively, some post-modeling methods are tailored to specific types of models, such as deep learning or hybrid architectures (*model-specific approaches*), allowing them to leverage internal model details. Let us now focus on a few general model-agnostic methods.

Textual justification uses natural language generation models to provide explanations in the form of phrases or sentences, helping users understand the model's predictions.

Visualization techniques analyze how a model captures hidden patterns during training or how it makes predictions during testing. By inspecting model structures, various visualizations can be generated from inputs and outputs. This method can also be combined with other explanation approaches to improve interpretability.

The *simplification* approach is perhaps the broadest category of model-agnostic post-modeling methods. It builds a simpler surrogate model that approximates the complex model's behavior while minimizing complexity and maintaining comparable performance. However, oversimplification can lead to a loss of ac-

curacy.

Feature relevance methods evaluate the model's internal workings by assigning importance scores to input variables. These scores quantify the influence or sensitivity of each feature, revealing which inputs the model relies on most when making predictions. This approach is often combined with visualization techniques to present the results in an accessible way.

One of the most well-known and commonly used feature relevance scoring is the SHapley Additive exPlanation (SHAP) values, which rely on the Shapley values from cooperative game theory and which I will discuss in more detail in the following sections.

3.2.2. XAI Methods Categorized by Scope of Explanation

Hassija et al. propose an alternative categorization system in [5], where one dimension of classification is based on the scope of the explanation.

Global interpretability refers to methods that aim to explain the overall behavior of a model, generating insights that apply across all possible inputs. These approaches seek to uncover the general logic or structure of the model and typically include techniques such as model simplification (e.g., model extraction) and feature-based analyses, like feature importance or feature interaction.

In contrast, *local interpretability* focuses on explaining individual predictions by examining how specific input features influence the outcome for a given instance. In such models, each feature is associated with a weight, and predictions are typically formed through a weighted combination of input features and a bias term. This structure allows for intuitive understanding of feature impact by isolating the effect of a single variable while holding others constant. Notable examples of local interpretability methods include various implementations of Shapley values or LIME (Local Interpretable Model-agnostic Explanations).

4. Shapley Values in Machine Learning

The application of Shapley values for explaining prediction models was initially introduced by Strumbelj and Kononenko in [15] and [17]. Their goal was to develop a model-agnostic, post-hoc explanation method that provides local interpretations by assigning contribution scores to individual features. To ensure broad applicability across different model types, their approach treats the model as a black box, analyzing how variations in input feature values affect the output predictions.

The following notations and assumptions will be used in this section. Let $\mathcal{X} = [0, 1]^n$ be our feature space (so the features $X_1, \ldots, X_n$ are normalized), and let Y be the target variable, and $\{y_i; x_{i,1}, \ldots, x_{i,n}\}_{i=1}^{M}$ a data set of M instances. The function $f : \mathcal{X} \to \mathbb{R}$ represents the model that is used to predict the value of the target variable for an instance $x \in \mathcal{X}$.

We are interested in how a specific value of a feature influences the model's prediction. To measure this, let us see the following metric [17]:

Definition 4.1. *For an $x = (x_1, \ldots, x_n)$ instance we define the **situational importance** of $X_i = x_i$ as $\psi_i(x) = f(x_1, \ldots, x_n) - \mathbb{E}[f(x_1, \ldots, X_i, \ldots, x_n)]$.*

With this definition we assign, to a specific value of a feature, the difference between a prediction for an instance and the expected prediction for the same instance if the ith feature had not been known. If the situational importance is positive, then the actual value of the feature has a positive contribution, if it is negative, then it has a negative contribution, and if it is 0, it has no contribution.

Naturally, the expected value of X_i should be approximated (or computed, if the feature's domain is finite), so we have to perturb the values of the ith feature, while the values of other input features remain fixed, and then average the predictions. The ultimate contribution of a feature in a prediction comes from the aggregation of its situational importances across all instances.

In additive models, where we assume no interactions between the features, the contribution of a feature is independent of the values of the others. For ex-

ample, in the linear regression model, where

$$f(x) = \beta_0 + \beta_1 x_1 \ldots + \beta_n x_n,$$

the metric is equivalent to the difference between what a feature contributes when its value is x_i and what it is expected to contribute:

$$\psi_i^{add}(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n - \beta_0 - \beta_1 x_1 - \ldots - \mathbb{E}[\beta_i X_i] - \ldots - \beta_n x_n = \\ = \beta_i x_i - \mathbb{E}[\beta_i X_i].$$

The situational importance can be easily interpreted, and it is very useful in the field of additive models, but if we want to take into account the interactions between features, we need to extend the method.

Definition 4.2. *We define the* ***contribution of a subset of feature values*** *as*

$$\Delta_Q(x) = f_Q(x) - f_{\{\}}(x), \text{ where}$$
$$f_Q(x) = \mathbb{E}[f(x)|X_i = x_i, \forall i \in Q], Q \subseteq [n] \text{ and } f_{\{\}} = \mathbb{E}[f(x)].$$

This is the change in prediction caused by observing the values of a certain subset of features (Q) for some instance $x \in \mathcal{X}$.

We can also look at the contribution of Q as the sum of interactions across all its subsets:

$$\Delta_Q(x) = \sum_{W \subseteq Q} I_W(x), \tag{1}$$

from which we can uniquely determine the *interactions* in Q:

$$I_Q(x) = \Delta_Q(x) - \sum_{W \subset Q} I_W(x). \tag{2}$$

For defining the contribution of one feature value, we should consider the case where we add feature i to all subsets in $[n]\backslash\{i\}$:

$$\phi_i(x) = \sum_{Q \subset [n]\backslash\{i\}} \frac{I_{Q\cup\{i\}}(x)}{|Q|+1}. \tag{3}$$

We can think about our model as a coalitional game in each instance, where the players are the feature values and the characteristic function is $\Delta_Q(x)$. Since $\Delta_{\{\}}(x) = 0$ and $\Delta_Q(x) \leq \Delta_R(x) \, \forall Q \subset R$, this is indeed a valid characteristic function. In this coalitional game, the concept of feature value contributions–defining the role of each feature value in the prediction–leads us to the same payoff formulation as Shapley values.

Theorem 4.1.

$$\begin{aligned}\phi_i(x) &= \sum_{Q \subseteq [n]\backslash\{i\}} \frac{|Q|!(n-|Q|-1)!}{n!}(\Delta_{Q\cup\{i\}}(x) - \Delta_Q(x)) = \\ &= \sum_{Q \subseteq [n]\backslash\{i\}} \frac{|Q|!(n-|Q|-1)!}{n!}(f_{Q\cup\{i\}}(x) - f_Q(x))\end{aligned}$$

Proof. Let us rewrite the recursive definition in (2) by induction:

$$I_Q(x) = \sum_{W \subseteq Q} (-1)^{|Q|-|W|} \Delta_W(x),$$

from which (3) is:

$$\phi_i(x) = \sum_{Q \subset [n]\backslash\{i\}} \frac{\sum_{W \subseteq Q\cup\{i\}} (-1)^{|Q\cup\{i\}|-|W|} \Delta_W(x)}{|Q \cup \{i\}|}.$$

Every subset W of $[n]\backslash\{i\}$ appears once in the right side sum in every term, where it is the subset of the current subset Q.

Let us first calculate the times of appearances for $\Delta_{V\cup\{i\}}(x), V \subseteq [n], i \notin V$. The sign of its terms depends on the parity of $|Q \cup \{i\}| - |V \cup \{i\}| = |Q| - |V|$. The total number of subsets that contain $V \cup \{i\}$ can be calculated taking all

combinations of $|Q| - |V|$ elements from the remaining $n - |V| - 1$. This can be written in the following form, and be expressed through the beta function:

$$M_{\Delta_{V\cup\{i\}}(x)} = \frac{\binom{n-|V|-1}{0}}{|V|+1} - \frac{\binom{n-|V|-1}{1}}{|V|+2} + \ldots + (-1)^{n-|V|-1}\frac{\binom{n-|V|-1}{n-|V|-1}}{n} = B(|V|+1, n-|V|) =$$
$$= \frac{\Gamma(|V|+1)\Gamma(n-|V|)}{\Gamma(n+1)} = \frac{|V|!(n-|V|-1)!}{n!}$$

Similarly, $\Delta_{V(x)}$ appears depending on the parity of $|Q \cup \{i\}| - |V|$, which is of course the opposite of the previous one. Apart from the sign change, the sum stays the same:

$$M_{\Delta_V(x)} = -M_{\Delta_{V\cup\{i\}}(x)}.$$

Taking the sum on all subsets V the theorem can be proved, since:

$$\phi_i(x) = \sum_{V\subseteq[n]\setminus i} M_{\Delta_{V\cup\{i\}}(x)}\Delta_{V\cup\{i\}}(x) + M_{\Delta_V(x)}\Delta_{V(x)} =$$
$$= \sum_{V\subseteq[n]\setminus i} \frac{|V|!(n-|V|-1)!}{n!}\Delta_{V\cup\{i\}}(x) - \frac{|V|!(n-|V|-1)!}{n!}\Delta_{V(x)}$$

□

As we have seen in Section 2, the Shapley values have a lot of practical attributes and they provide a somewhat fair payoff among the players, which is now a fair scoring of the feature values' importance in a single prediction.

Remark 4.1. *Let us recall the efficiency property of Shapley values, now meaning that if we sum over all feature values, we get back the prediction for the given instance $f(x)$.*

4.1. Sampling Method for Approximating Shapley Values

It has been established that our machine learning model can be interpreted as a coalitional game, in which the players correspond to the feature values and the

value function represents the contribution of a subset of features to the prediction for a given instance. Furthermore, it has been shown that the contribution of an individual feature is equivalent to the Shapley value from cooperative game theory.

However, computing the Shapley values exactly requires knowledge of both the joint distribution of the features and the predictor function f. In practical applications, such complete information is rarely available.

As a result, the best achievable approach is to estimate the expected value of f using the sample mean of predictions. This estimation procedure entails computing the mean prediction across all possible feature subsets, which leads to an exponential computational complexity.

In the following section, various estimation algorithms will be presented, beginning with the sampling approach proposed by the original authors [17].

Let us remember the *random arrival formula* from Theorem 2.1, which was an equivalent form of the Shapley values:

$$\phi_i(x) = \frac{1}{n!} \sum_{\pi \in S_n} \Delta_{\pi\{1,\dots,k\}}(x) - \Delta_{\pi\{1,\dots,k-1\}}(x),$$

where S_n is the set of all permutations of instances $\{1, \dots, n\}$ and $\pi(k) = i$. Now we could sample permutations from S_n randomly, but this does not solve our problem yet, because the computational cost of $\Delta(x)$ is still exponential.

Another possible simplification is to express the contributions in the instance space $\mathcal{X}$ and to limit ourselves to such distributions of instances p that individual features are distributed independently (efficiency is shown in [16]). This results in the following reformulation:

$$\begin{aligned} \Delta_{\pi\{1,\dots,k\}}(x) &= f_{\pi\{1,\dots,k\}}(x) - f_{\{\}}(x) = \\ &= \sum_{\omega \in \mathcal{X}, \omega_j = x_j, \forall j \in \pi\{1,\dots,k\}} p(\omega) f(\omega) - \sum_{\omega \in \mathcal{X}, \omega_j = x_j \, \forall j} p(\omega) f(\omega) = \\ &= \sum_{\omega \in \mathcal{X}} p(\omega)[f(\omega | \omega_j = x_j, j \in \pi\{1, \dots, k\}) - f(\omega)]. \end{aligned}$$

Similarly,

$$\Delta_{\pi\{1,\ldots,k-1\}}(x) = \sum_{\omega \in X} p(\omega)[f(\omega|\omega_j = x_j, j \in \pi\{1,\ldots,k-1\}) - f(\omega)],$$

so the Shapley value of feature i in instance x can be expressed as:

$$\phi_i(x) = \frac{1}{n!} \sum_{\pi \in S_n} \sum_{\omega \in X} p(\omega)[f(\omega|\omega_j = x_j, j \in \pi\{1,\ldots,k\}) - f(\omega|\omega_j = x_j, j \in \pi\{1,\ldots,k-1\})].$$

Based on this, the sampling procedure will be the following, including random sampling of instances and permutations:

Algorithm 1 Sampling Algorithm

Require: Feature vector x (instance), feature index j, predictor function f, number of samples m

Ensure: Estimated Shapley value $\hat{\phi}_j(x)$

1: **for** $k = 1$ to m **do**
2: Sample a random permutation $\pi \in S_n$ of the feature indices
3: Sample an instance ω from the data distribution (according to p)
4: Let $S = \{i \in [n] : \pi(i) < \pi(j)\}$
5: Construct two instances:
- x_S: instance ω where features in S are set to values from x
- $x_{S \cup \{j\}}$: instance ω where features in $S \cup \{j\}$ are set to values from x

6: Compute marginal contribution:

$$V_k = f(x_{S \cup \{j\}}) - f(x_S)$$

7: **end for**
8: **return** $\hat{\phi}_j(x) = \frac{1}{m} \sum_{k=1}^{m} V_k$

The sampling method described above yields an estimator $\hat{\phi}_i(x)$ that is *unbiased* and *consistent* for the true Shapley value $\phi_i(x)$. Moreover, under the Central Limit Theorem, as the sampling size m tends to ∞, $\hat{\phi}_i(x)$ is approximately normally distributed around $\phi_i(x)$, with variance $\frac{\sigma^2}{m}$, where σ^2 denotes the sample variance of the marginal contributions and m is the number of samples.

It is important to note that, if the number of features is small, then of course

the total number of distinct permutations (or coalitions) is also small. In such cases, even taking many samples does not improve the approximation meaningfully, since the underlying distribution of marginal contributions is discrete and of limited support. Therefore, in small feature spaces, the exact Shapley values can often be computed directly and sampling-based approximation is unnecessary or even inappropriate.

Furthermore, the special case where the model function f is constant (i.e., $f(x) \equiv c$ for all x) is naturally handled in this framework. In such cases, all marginal contributions are zero, resulting in a sample variance $\sigma^2 = 0$, and hence the estimator $\hat{\phi}_i(x)$ has zero variance for any m. Consequently, $\hat{\phi}_i(x) = \phi_i(x) = 0$ exactly, and the CLT holds trivially.

What is the time complexity of this method? When explaining an instance, the sampling process has to be repeated for each of the n feature values, so the time complexity is $O(n\mathcal{T}(\mathcal{X}))$, where function $\mathcal{T}(\mathcal{X})$ describes the instance classification time of the model on $\mathcal{X}$ (for most machine learning models $\mathcal{T}(\mathcal{X}) \leq n$).

Several strategies have been proposed to improve the efficiency of this baseline algorithm. One such approach involves adapting the number of samples m allocated to each feature based on the observed variability of its marginal contributions [17]. Another approach proposed in [11] builds reproducing kernel Hilbert spaces (RKHS) on the group of permutations using Kendall and Mallows kernels. In that space the so called kernel herding method is used, which is a greedy optimisation process selecting points for maximum separation, while also converging to the expected distribution p, resulting in a quasi-Monte Carlo algorithm. The rate of the convergence for a general Monte Carlo method is $O(\frac{1}{\sqrt{m}})$, which is improved by this method to $O(\frac{1}{m})$.

4.2. SHAP Values

An alternative approach and interpretation of Shapley values is the widely used SHAP (Shapley Additive Explanation) method. It was introduced by Lundberg and Lee in 2017 [7], with the primary objective of unifying existing model explanation techniques within a general theoretical framework.

The SHAP approach can be understood as a *simplification method*, wherein a complex, black-box model is approximated by a simpler and more interpretable **explanation model**. Let the original predictive model be denoted by f, and the explanation model by g. For simplicity, SHAP employs binary *simplified input* vectors $x' \in \{0,1\}^n$, which indicate the presence or absence of each original feature. These simplified inputs are mapped back to the original input space via a function $x = h_x(x')$.

Under this formulation, the goal is to locally approximate the predictions of the original model using the explanation model, i.e.,

$$g(x') \approx f(h_x(x')).$$

Definition 4.3. *We call a model **additive feature attribution method**, if it has an explanation model which is structured as follows:*

$$g(x') = \phi_0 + \sum_{i=1}^{n} \phi_i(x) x'_i,$$

where x' is the simplified input feature and $\phi_i(x) \in \mathbb{R}$.

Since the sum of the Shapley values across all features exactly recovers the model prediction for a given instance, the method can be interpreted as an additive feature attribution model. This interpretation holds even when the Shapley values are approximated using the sampling-based method described in Section 4.1, where the binary variable x'_i indicates the presence or absence of feature i in the simplified input representation.

We can define three desirable properties that the explanation models of the additive feature attribution method class should satisfy:

1. *Local accuracy:* $f(x) = g(x')$.

   When approximating the original model f at a specific input x, the explanation model is required to exactly reproduce the output of f at that point.

2. *Missingness:* $x'_i = 0 \Rightarrow \phi_i(x) = 0$.
   If a feature is absent in the simplified input x', it is assumed to be missing from the original input. The missingness property states that such features must have zero contribution in the explanation model.
3. *Consistency:* for any two models f and f', if for all inputs z',
   $f'(h_x(z')) - f'(h_x(z'|z'_i = 0)) \geq f(h_x(z')) - f(h_x(z'|z'_i = 0))$,
   then $\phi_i^{f'}(x) \geq \phi_i^f(x)$.
   This property ensures that if the model changes in such a way that the marginal contribution of a feature increases or remains unchanged (regardless of other features), then the feature's assigned attribution should not decrease.

Theorem 4.2. *For every model f and mapping $h_x(z)$ there is only one possible explanation model g that follows the additive feature attribution definition and satisfies the previous three properties:*

$$\phi_i^f(x) = \sum_{z' \subseteq x'} \frac{|z'|!(n - z' - 1)!}{n!}[f(h_x(z')) - f(h_x(z'|z'_i = 0))],$$

where $|z'|$ is the number of non-zero entries in z', and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x'.

Proof. Similar to its game theoretic equivalent (Theorem 2.1). □

In Section 4.1, we have already seen that feature value contributions lead us to the Shapley values with $f_Q(x) = \mathbb{E}[f(x)|X_i = x_i, \forall i \in Q]$. Let us choose the mapping function in the former theorem as follows:

- Let S be the non-zero indices in z'
- Let z_S be the version of z with missing values in features $i \notin S$
- $h_x(z') \coloneqq z_S$.

Definition 4.4. *With this mapping function $f(h_x(z')) = f(z_S) \approx E[f(z)|z_S]$, and the values defined with this in Theorem 4.2 are the **SHAP values**.*

This definition is sufficiently general to encompass other additive feature attribution methods, such as LIME (Local Interpretable Model-agnostic Explanations) and DeepLIFT, while still satisfying the three desirable properties outlined above: local accuracy, missingness, and consistency.

If we assume feature independence when approximating conditional expectations, then SHAP values can be estimated directly using the Shapley sampling method. In the following sections, we will explore alternative approaches for estimating SHAP values, including both model-agnostic and model-specific methods.

4.2.1. Model-agnostic Approximation of SHAP Values

Kernel SHAP

Kernel SHAP is a model-agnostic method introduced in [7] for explaining individual predictions that combines elements from LIME and Shapley values. While LIME approximates a complex model f locally using a linear surrogate g, Kernel SHAP seeks to ensure that this approximation also respects the key theoretical properties unique to SHAP values (*local accuracy*, *missingness*, and *consistency*).

LIME formulates the explanation problem as a regularized regression in a simplified binary input space. In this setting, as before, simplified inputs z' are binary vectors indicating the presence or absence of features. A mapping $h_x(z')$ reconstructs the original input x from z, allowing the explanation model g to approximate the true prediction $f(h_x(z))$. LIME minimizes the following objective:

$$\arg\min_{g \in \mathcal{G}} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

where $\pi_x(z')$ is a local weighting kernel and $\Omega(g)$ is a regularization term controlling the complexity of g.

LIME's heuristic choice of kernel $\pi_x(z')$, loss function $\mathcal{L}$, and regularizer Ω do not generally yield Shapley-consistent attributions. This can result in vi-

olations of local accuracy or consistency, potentially leading to unintuitive or misleading feature importances in certain edge cases.

Kernel SHAP addresses this limitation by explicitly identifying the parameter choices that guarantee consistency with Shapley values. Specifically, it derives the **Shapley kernel**:

- **Weighting kernel**:
$$\pi_x(z') = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)},$$
- **Loss function**: $\sum_{z' \in Z} \left(f(h_x(z')) - g(z')\right)^2 \pi_x(z')$,
weighted squared error between $f(h_x(z))$ and $g(z')$,
- **Regularization term**: $\Omega(g) = 0$.

By enforcing these settings, Kernel SHAP solves LIME's regression-based formulation in a way that satisfies the axioms underpinning Shapley values. For the proof of this statement see [7].

In practical applications, however, Kernel SHAP does not evaluate the full power set of simplified inputs due to computational constraints, as the number of subsets grows exponentially with the number of features. Instead, it relies on sampling a subset of these inputs to approximate the SHAP values, in order to scale to moderately high-dimensional problems while maintaining the desirable theoretical properties of the Shapley framework.

4.2.2. Model-specific Approximation of SHAP Values

Several model-specific SHAP methods have been developed to efficiently compute SHAP values by leveraging the structure of particular model classes. For example Linear SHAP [7] provides an exact computation of SHAP values for linear models by directly using model coefficients and feature distributions, Deep SHAP [7] extends SHAP value estimation to deep neural networks by combining the principles of SHAP with the DeepLIFT algorithm, while Max SHAP [7] is tailored for models such as certain max-pooling neural network layers or max-based decision rules. In the following I will talk about a method, also introduced by Lundberg et. al in 2018 [8], designed for tree-based models: Tree SHAP.

Tree SHAP

Tree SHAP is a model-specific method designed to efficiently compute exact Shapley values for tree-based models, such as decision trees, random forests, and gradient-boosting methods. Unlike Kernel SHAP, which is model-agnostic and relies on sampling to approximate Shapley values, Tree SHAP leverages the internal structure of tree ensembles to calculate Shapley values in polynomial time.

Tree SHAP exploits the structure of decision trees to reduce the complexity of exact SHAP value computation from $O(TL2^n)$ to $O(TLD^2)$, where T is the number of trees, L is the maximum number of leaves per tree, n is the number of features, and D is the maximum depth of any tree. This exponential reduction in computational complexity enables the exact calculation of SHAP values for complex models. Both versions of the algorithm can be found in [8].

The first straightforward recursive algorithm estimates the conditional expectations $\mathbb{E}[f(x) \mid X_i = x_i \, \forall i \in Q]$ for feature subsets Q, by traversing the tree nodes, using node values, thresholds, feature splits, and sample cover counts to weigh the contributions. This direct method has exponential complexity ($O(TL2^n)$).

To overcome this, the algorithm introduces a polynomial-time method with complexity $O(TLD^2)$. The key idea is to recursively track the proportions of all possible feature subsets flowing through each node, while accounting for subset sizes to correctly weight their contributions in the Shapley formula.

This is achieved via two complementary procedures: grows subsets along the tree branches, and reverses previous extensions to maintain correct weighting when features appear multiple times on a path. The algorithm efficiently aggregates these weighted contributions to compute exact SHAP values for all features, leveraging the tree structure to avoid the exponential blowup.

5. Numerical Experiments

5.1. Performance of Shapley Value Estimators on Synthetic Data

To evaluate the performance of various Shapley value estimation methods, I conducted a series of experiments using synthetic data generated in the R programming language. The implementations of the original sampling-based algorithm was utilized via the `iml`[1] and `fastshap`[2] packages, while the SHAP-based algorithms were utilized via the `kernelshap`[3] and `treeshap`[4] packages. All scripts from this section can be found in the GitHub repository of this thesis[1].

In Experiments A and B, the focus was on models with known, additive predictor functions, for regression and classification tasks. This setting allows for the analytical computation of theoretical Shapley values, which served as a benchmark to assess the accuracy of the other models.

In contrast, Experiments C involved more complex tree-based models, where theoretical Shapley values are not easily obtainable. In these cases, the approximations produced by the methods were compared against the exact Kernel SHAP values.

5.1.1. Experiment A: Linear Regression

This experiment involved a regression task using a linear regression model from the `lm` package. For each simulation, a synthetic dataset of $n = 100$ observations was generated, in twenty iterations. Each dataset consisted of $M = 5$ features independently drawn from a standard uniform distribution. The target variable was generated according to the following linear model:

$$Y = X_1 + 2X_2 - X_3 + \epsilon, \epsilon \sim N(0, 0.1^2).$$

[1] `https://github.com/giuseppec/iml`
[2] `https://github.com/bgreenwell/fastshap`
[3] `https://github.com/ModelOriented/kernelshap`
[4] `https://github.com/ModelOriented/treeshap`

Given the additive structure of the linear model, the exact Shapley values for each feature i and instance x can be computed as:

$$\phi_i^{Exact}(x) = \beta_i[x_i - \mathbb{E}(X_i)] = \beta_i[x_i - 0.5].$$

An empirical approximation to the expectation can be computed using the sample mean:

$$\phi_i^{Exhaustive}(x) = \beta_i[x_i - \overline{X}_i)].$$

The sampling-based Shapley values were computed using the `iml` package, with low sample sizes $m = 25, 50, 100$.

Another implementation of the sampling method, in the `fastshap` package, was used to approximate SHAP values for each instance. This method was applied with the same sample sizes: $m = 25, 50, 100$.

I also utilized the Kernel SHAP method, implemented in the `kernelshap` package, with the whole sample set.

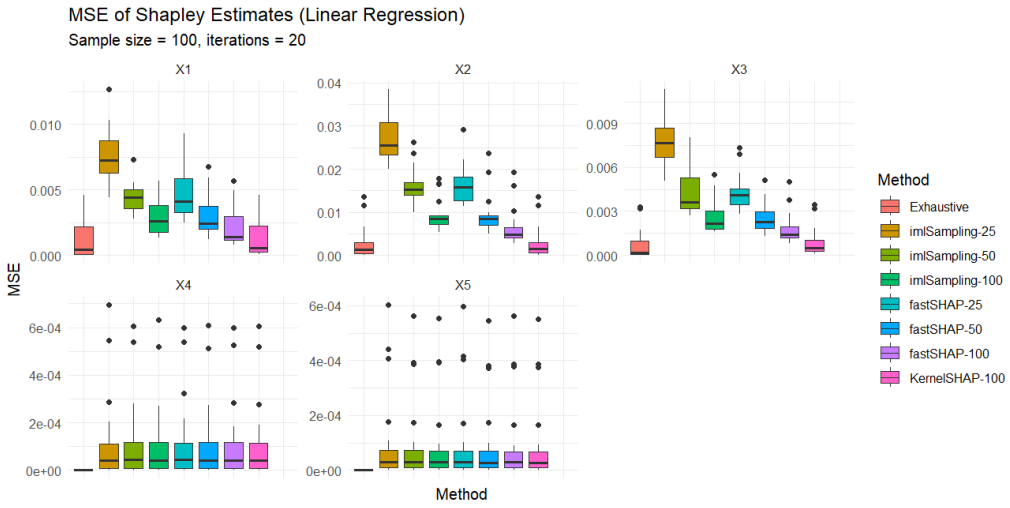


Figure 1: MSE comparison of estimation methods for a linear regression model

For each method, I computed the mean squared error (MSE) between the estimated Shapley values and the theoretical values. The MSE was calculated for each feature i across all instances. To ensure robustness, the simulation was

repeated twenty times. The resulting MSE values from each simulation were visualized using boxplots, which are presented in Figure 1.

From the results, it is evident that for the meaningful features X_1, X_2, X_3, the `fastshap` implementation consistently outperforms the `iml` package across all three sample sizes in pairwise comparisons. The best approximation is achieved by Kernel SHAP, which is essentially equivalent to the exhaustive method and provides an almost perfect approximation of the theoretical Shapley values.

5.1.2. Experiment B: Logistic Regression with Log-Odds

In this experiment, I applied a logistic regression model for a binary classification task, using synthetic data. The data generation process and model fitting were similar to those in Experiment A, but the focus here was on the log-odds scale, which provides the predicted log-odds of the response variable, rather than the probabilities.

The data was generated as follows: The feature matrix X was composed of $n = 100$ instances with $M = 5$ features sampled from a uniform distribution. The logits, or the log-odds, were computed using the true model coefficients:

$$\text{logits} = X_1 + 2X_2 - X_3.$$

The probabilities for the binary outcome y were obtained using the logistic function:

$$P(Y = 1) = \frac{1}{1 + e^{-\text{logits}}}.$$

The response variable y was then sampled from a Bernoulli distribution with success probability $P(Y = 1)$. The logistic regression model was fitted to the data using the `glm` function, and the Shapley values were calculated using different methods on the log-odds scale.

The theoretical Shapley values for the log-odds were computed using the known coefficients of the logistic regression model. For each feature i, the Shap-

ley value was calculated as:

$$\phi_i^{Exact}(x) = \beta_i[x_i - \mathbb{E}(X_i)] = \beta_i[x_i - 0.5],$$

where $\mathbb{E}(X_i)$ is the mean of the feature X_i, which is 0.5 for a uniform distribution.

The exhaustive Shapley values were computed again by replacing the expected value of each feature with its sample mean across the training data.

In addition to the exact and exhaustive methods, Shapley values were estimated using both the sampling-based algorithms from `iml` and `fastshap` on the log-odds scale, each evaluated at three different sample sizes: $m = 25, 50, 100$, and the Kernel SHAP algorithm.

For every method, Shapley values were again computed for all instances in each dataset. The mean squared error (MSE) was then calculated for each feature across all instances, using the exact values as the ground truth. This process was repeated across the twenty simulations, and the resulting MSEs were visualized using boxplots, as shown in Figure 2.

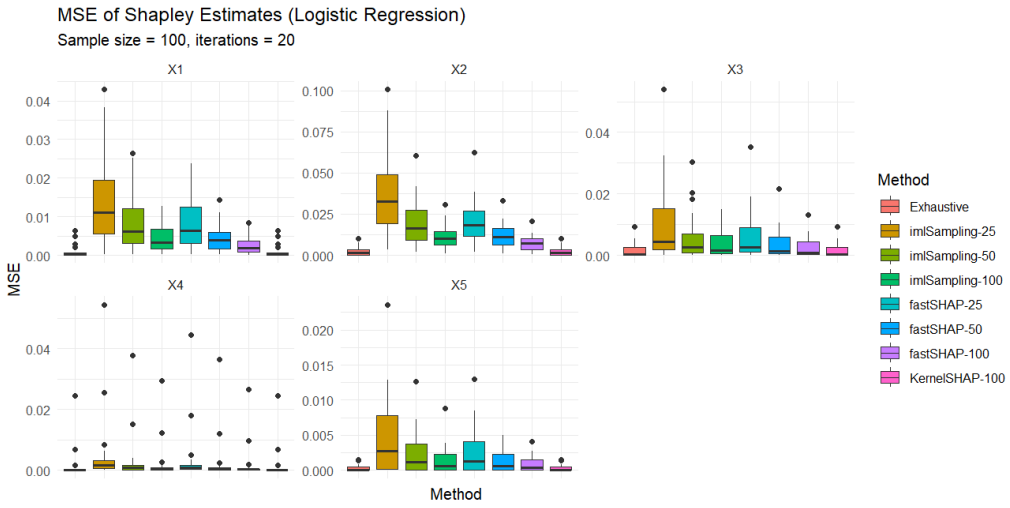


Figure 2: MSE comparison of estimation methods for a logistic regression model

The results are similar to those observed in the regression task of Experiment A; however, there are features for which the differences are smaller (e.g., X_3)

and others where they are larger (e.g., X_2). The variance also varies accordingly. Additionally, a noticeable difference is observed for feature X_5, which does not contribute to the predictions.

5.1.3. Experiment C: Regression with Tree-based Models

In this experiment, I investigated more complex, non-linear regression models, for which theoretical Shapley values are no longer analytically tractable. Specifically, I considered two model types: a random forest and an eXtreme Gradient Boosting (XGBoost) regressor. These models were trained on synthetic data, and the Shapley values were estimated using the approximation methods used before: the sampling method, fastshap and Kernel SHAP, and additionally with Tree SHAP.

The data was generated the same way as before, and the models were fitted using the `xgboost`[5] and `caret`[6] packages. Since these models introduce non-linearities and interactions not present in a linear regression, exact or exhaustive computation of Shapley values was not feasible. Instead, I used the Kernel SHAP approximation as a benchmark reference for evaluating the accuracy of the other SHAP estimation methods. The metrics stayed the same as before.

The resulting MSE values, calculated per feature across all instances, are shown in Figure 3, according to the models.

Sampling methods exhibit similar performance as previously observed for features X_1, X_2, and X_3, with the variance of the fastshap implementation being substantially lower. The scale of errors remains consistent across both Random Forest and XGBoost models. For features X_4 and X_5, noticeable differences in performance are again evident. Tree SHAP performs exceptionally well overall, although it tends to produce larger errors for the non-important features X_4 and X_5, comparable to the error magnitude of fastSHAP-100.

[5] `https://github.com/dmlc/xgboost`
[6] `https://github.com/topepo/caret`

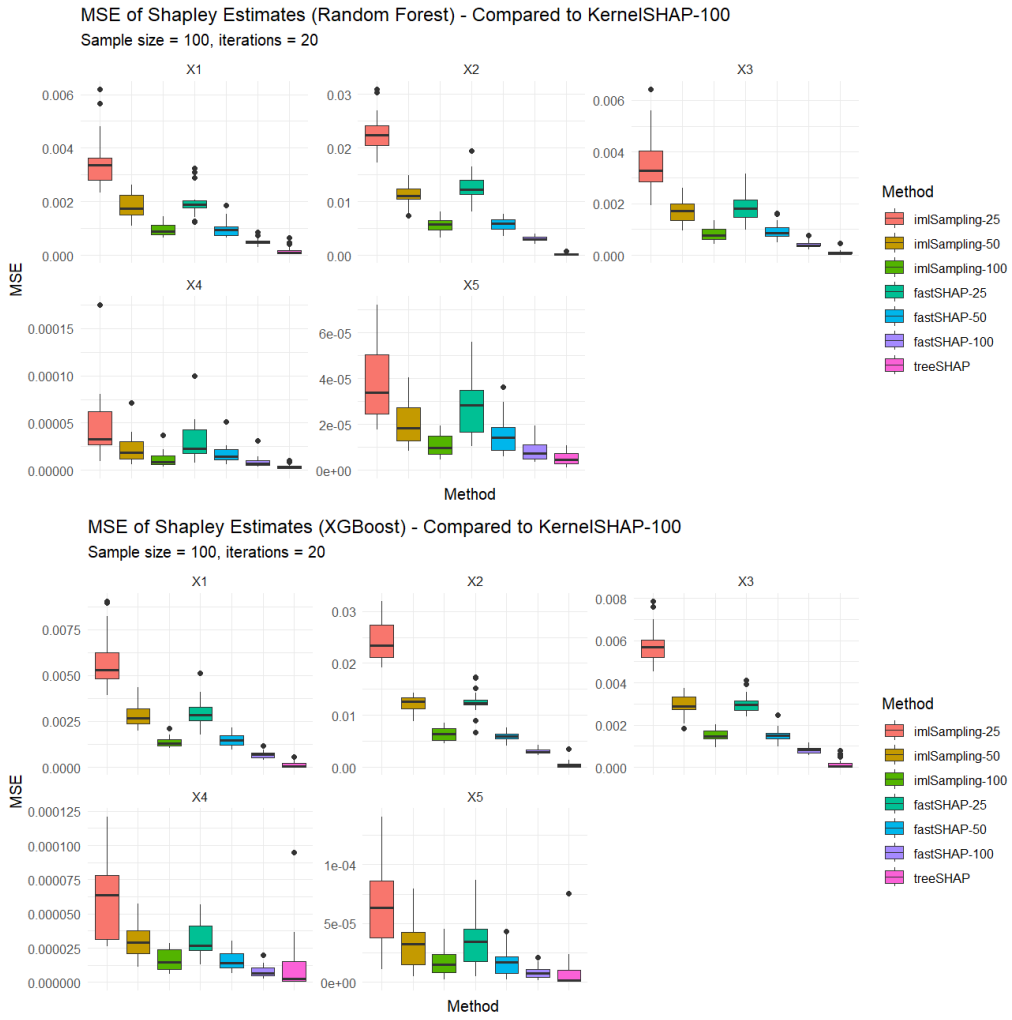


Figure 3: MSE comparisom of estimation methods, compared to the Kernel SHAP method, for a Random Forest and an XGBoost regression model

It is also important to note that for tree-based methods, the Tree SHAP algorithm is significantly faster than the model-agnostic Kernel SHAP method (as discussed in the previous section). Therefore, Tree SHAP should be our preferred choice, especially for larger datasets and models with many features.

5.2. Case Study: Fire Hazard Estimation

As part of a recent project, I contributed to the development of wildfire hazard estimates in the United Kingdom, a region that has traditionally experienced relatively low wildfire activity. However, in recent years, particularly due to rising temperatures and prolonged dry periods, wildfires have become more frequent and severe [2]. The project was based on the results on wildfire risk in California[4].

To analyze this shift in the UK, we worked with openly available weather and satellite data spanning the wildfire seasons (May to September) from 2019 to 2023. The core of our analysis involved modeling wildfire hazard as a classification problem, using XGBoost models to make predictions.

For a more comprehensive understanding of wildfire risk and its causes in the UK, my contribution is twofold now: first, to present and evaluate the results of our previous work by applying SHAP values. And second, to extend the modeling framework by training a neural network and interpret these new results with SHAP values to identify the most influential factors driving wildfire risk.

5.2.1. Dataset Design

During the dataset building we gathered openly available data as explanatory variables such as weather and satelite data, land cover or roads, settlements. Our target variable contained the burnt pixels on a given timestamp.

The spatial resolution and temporal extent of our data were determined based on the availability of data and taking into account our computational capacity. The resolution of the variables was adjusted to the target variable, burnt areas, using interpolation or aggregation when needed.

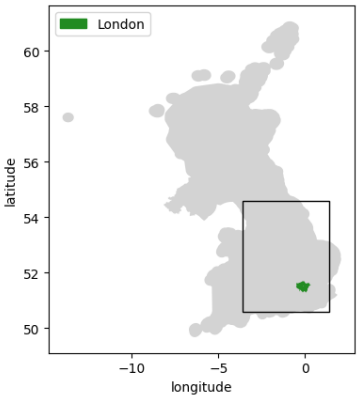


Figure 4: The studied area

In Table 1, the parameters of the final dataset are presented. The data is fully available through the GitHub repository associated with this thesis [1]. Tables 2 and 3 list the variables of the dataset along with their original attributes and sources.

Table 1: Spatial and temporal attributes of the dataset

	Extent	Resolution
Temporal	2019-2022, May-September; 2023 May-July	weekly
Spatial	Latitude: (50.6°) – (54.6°) Longitude: (−3.6°) – (1.4°)	$300 \times 300\ m$

Most of the variables originate from the Land Copernicus and Climate Copernicus portals, which provide openly accessible data for Europe. Spatial information, such as distances from major roads or residential areas, was derived using Mapbox, while elevation data was obtained from NASADEM.

Table 2: Static variables of the dataset, with their original attributes

Variable Name	Abbr.	Unit of Measurement	Spatial res.
Elevation[1]	EL	m	$30m$
Distance from railways[2]	rail_dist	km	$300m$
Distance from main roads[2]	main_dist	km	$300m$
Distance from minor roads[2]	small_dist	km	$300m$
Distance from residential areas[2]	city_dist	km	$300m$

[1] `https://portal.opentopography.org/datasetMetadata?otCollectionID=OT.032021.4326.2`

[2] `https://docs.mapbox.com/data/tilesets/reference/mapbox-streets-v8/`

Table 3: Dynamic variables of the dataset, with their original attributes

Variable Name	Abbreviation	Unit of Measurement	Spatial resolution	Temporal resolution
Burnt Area (*target variable*)[1]	BA_bin	m	300 m	daily
Normalised difference vegetation index[2]	NDVI	–	300 m	10 daily
Fraction of absorbed photosynthetically active radiation[3]	FAPAR	–	300 m	10 daily
Land surface temperature[4]	LST	$°C$	5 km	10 daily
Soil water index[5]	SWI	%	1 km	daily
Mean temperature (2m height)[6]	MT	$°C$	0.1°	daily
Max. temperature (2m height)[6]	MXT	$°C$	0.1°	daily
Precipitation amount[6]	PA	mm	0.1°	daily
Relative humidity (2m height)[6]	RH	%	0.1°	daily
Wind speed (10m height)[6]	WS	$\frac{m}{s}$	0.1°	daily
Land cover[7]	LC	factor	300 m	yearly

[1] https://land.copernicus.eu/en/products/vegetation/burnt-area-v3-1-daily-300m

[2] https://land.copernicus.eu/en/products/vegetation/normalized-difference-vegetation-index-300m-v1.0

[3] https://land.copernicus.eu/en/products/vegetation/fraction-of-absorbed-photosynthetically-active-radiation-v1-0-300m

[4] https://land.copernicus.eu/en/products/temperature-and-reflectance/10-daily-land-surface-temperature-daily-cycle-global-v1-0-5km

[5] https://land.copernicus.eu/en/products/soil-moisture/daily-soil-water-index-euro

[6] https://cds.climate.copernicus.eu/datasets/insitu-gridded-observations-europe?tab=overview

[7] https://cds-beta.climate.copernicus.eu/datasets/satellite-land-cover?tab=overview

5.2.2. Modeling

The modeling was conducted in Python, utilizing widely used scientific libraries including `NumPy`, `pandas`, `scikit-learn` and `XGBoost`. The boosting model presented here was developed during the previous year's project, whereas the neural network has been newly trained for this work. All scripts used for model development and evaluation are available in the accompanying GitHub repository[1] as well as the fine-tuned models.

Both models were trained and validated using data from 2019 to 2022, and subsequently tested on data from July 2023.

XGBoost

As part of the *data preprocessing* phase, Synthetic Minority Over-sampling Technique (SMOTE) was employed to mitigate class imbalance by oversampling the minority class of burnt pixels, which initially comprised $3,246$ samples. To represent the majority class, $8,000$ non-burnt pixels were randomly selected from each of the four years in the training dataset.

Model training was conducted using the `XGBoostClassifier` from the `XGBoost` library. For the purpose of *hyperparameter optimization* we utilized the `Randomized SearchCV` function from the `scikit-learn` library, which samples combinations of hyperparameters at random from predefined ranges. In our case, 250 unique configurations were evaluated. While this approach does not guarantee convergence to a global optimum, it significantly reduces computational overhead compared to exhaustive grid-based methods. The number of boosting rounds (i.e., gradient-boosted decision trees) was fixed at 200 across all runs to ensure comparability.

The final model was selected based on its performance measured by the area under the receiver operating characteristic curve (ROC-AUC) evaluation metric. When tested on an unseen dataset corresponding to July 2023, the model achieved a ROC-AUC score of *0.8224*, indicating strong discriminative ability in distinguishing between burnt and non-burnt pixels. The parameters of this

best model can be found in Table 4.

Table 4: Hyperparameters of the fine-tuned XGBoost model

Parameter	**Value**	**Meaning**
`tree method`	'approx'	specify which training boosted tree method to use
`reg_lambda`	3.5	L2 regularization term on weights
`reg_alpha`	0.5	L1 regularization term on weights
`max_depth`	5	maximum tree depth for base learners
`learning_rate`	0.26	boosting algorithm learning rate per iteration
`grow_policy`	'depthwise'	tree growing policy
`booster`	'gbtree'	specify which booster to use

Neural Network

To reduce computational demands, the study area was restricted to the South-East England region, bounded by 50.97°–52.1° N latitude and 0°–1° E longitude. This region includes the eastern area of London, where one of the most significant grassland fires of the 2022 heatwave–the Wennington wildfire–occurred[7].

During *data preprocessing*, the same undersampling and oversampling strategies used for the boosting model were applied. However, a key enhancement in this approach was the inclusion of both temporal and spatial neighborhood context for each observation. Specifically, aggregated feature values were derived from the same week at neighboring spatial coordinates (using the arithmetic mean for numerical features and the mode for categorical ones), along with feature values from the same coordinates during the previous week.

[7] `https://en.wikipedia.org/wiki/Wennington_wildfire`

The model architecture consisted of a multilayer perceptron (MLP), implemented via the `MLPClassifier` class from the `scikit-learn` library. Model hyperparameters were selected heuristically and are summarized in Table 5.

The trained neural network was evaluated on the dataset from July 2023, achieving a ROC-AUC score of 0.7547.

Table 5: Hyperparameters of the MLPClassifier model

Parameter	**Value**	**Meaning**
`hidden_layer_sizes`	(100, 50)	number of neurons in each hidden layer
`activation`	'relu'	activation function for the hidden layers
`solver`	'adam'	the optimization algorithm used for training
`alpha`	0.001	L2 regularization term
`max_iter`	200	maximum number of training iterations (epochs)
`early_stopping`	True	training stops early if validation score doesn't improve for several epochs

5.2.3. Evaluating the Results with SHAP Values

Based on the experiments presented in Section 5.1, we can conclude that the SHAP estimations provide sufficiently accurate explanations. Therefore, SHAP values were used to interpret the decisions made by both the XGBoost and the Neural Network models. The analysis was conducted using the `shap`[8] package in Python, which includes implementations of both the Kernel SHAP and Tree SHAP algorithms, along with convenient visualization tools.

[8] `https://github.com/shap/shap`

Predictions were made with both models on the test set from July 2023, and the results were interpreted using SHAP values computed on a randomly sampled subset. Since the Tree SHAP algorithm, compatible with the XGBoost model, is significantly faster, here $100,000$ instances were sampled for each week, while in the case of the MLP model, Kernel SHAP was used on $\sim 1,000$ samples.

XGBoost

In Figure 5, we present beeswarm plots of the SHAP values for the XGBoost model across four weeks of July 2023. In each plot, every point represents the SHAP value of a specific feature value for a given instance. The color of the point reflects the feature's actual value, while the x-axis shows the SHAP value, indicating the feature's impact on the model output. Features are ranked by their overall importance, measured by the mean absolute SHAP value.

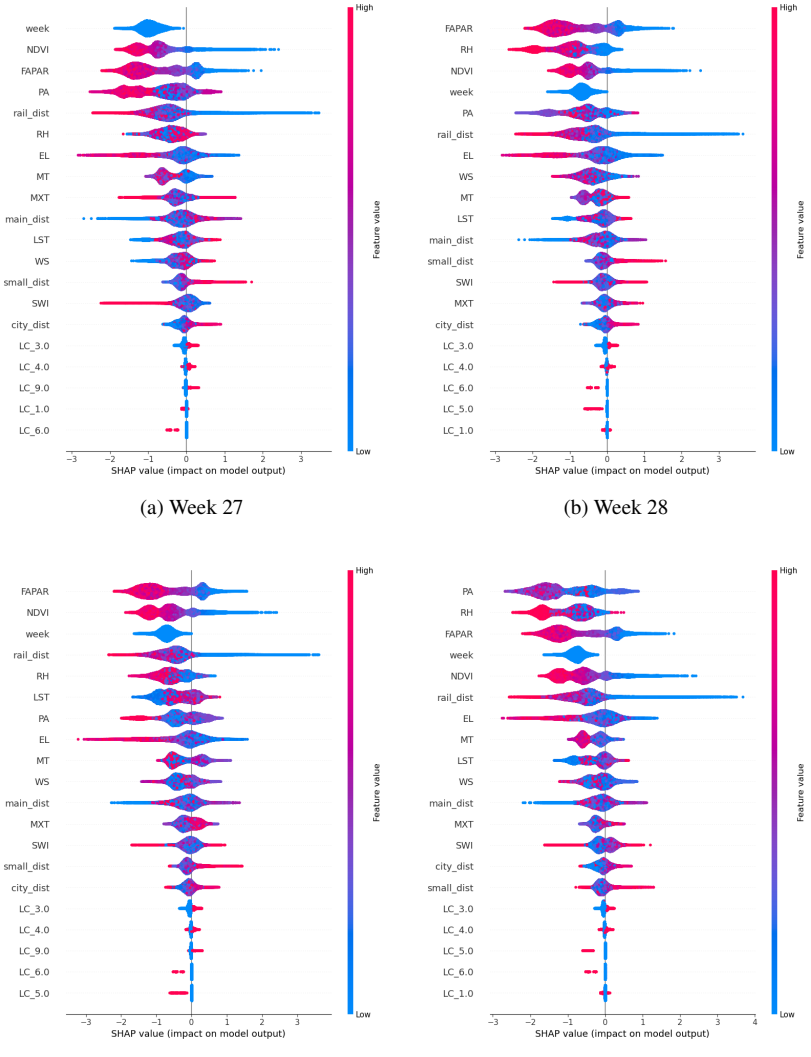
Although there are differences between the weekly plots, several consistent patterns emerge. Land cover variables typically show low importance, suggesting they play a limited role in fire hazard prediction. Interestingly, all weekly indices tend to push the prediction slightly toward zero, possibly indicating that fire risk increases in August compared to July.

Satellite-derived indices appear among the most important predictors. High values of FAPAR (fraction of absorbed photosynthetically active radiation), NDVI (normalised difference vegetation index), and RH (relative humidity) generally reduce predicted fire hazard, consistent with their association with vegetation health and moisture. In contrast, PA (precipitation amount), MT (mean temperature), and MXT (maximum temperature) show more ambiguous or inconsistent effects, possibly due to interactions or regional variability.

Elevation shows a clear negative relationship with fire risk–higher elevations tend to lower the hazard, which is intuitively reasonable given climate and vegetation changes with altitude.

Among distance-related features, distance from railways has the most notable

effect: the closer an area is to a railway, the higher the predicted fire hazard. This may reflect fire ignition risks associated with train traffic. Distance to main roads shows the opposite trend, though with a smaller average effect. Notably, proximity to minor roads and cities tends to increase hazard predictions, possibly due to increased human activity and ignition sources in these areas, bringing attention to suburban regions.



(a) Week 27

(b) Week 28

(c) Week 29

(d) Week 30

Figure 5: SHAP values for the XGBoost model for the weeks of July 2023

Neural Network

Figure 6 displays a beeswarm plot of the SHAP values corresponding to the neural network model's predictions for July 2023. Due to the computational intensity of Kernel SHAP compared to Tree SHAP, the analysis is based on approximately 1,000 randomly sampled instances.

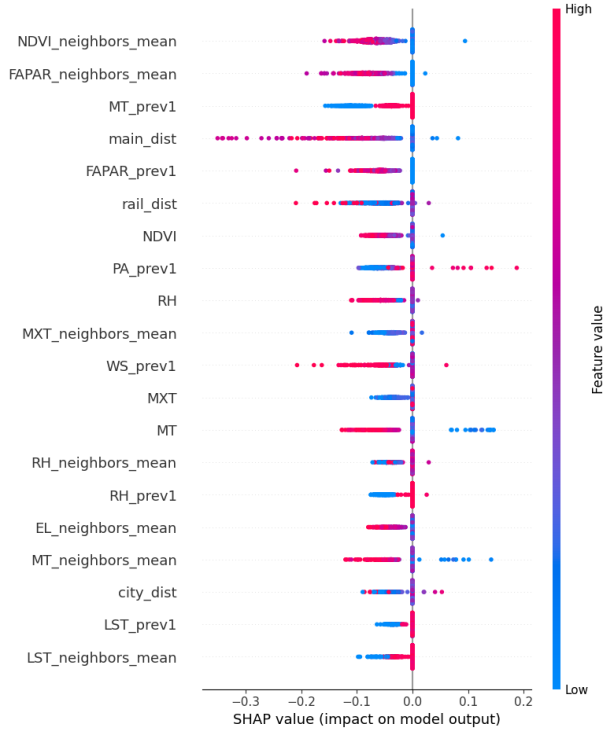


Figure 6: SHAP values for the MLP model for July 2023

Consistent with previous observations, land cover variables exhibit minimal importance and do not prominently appear in the plot, suggesting their limited contribution in this context. In contrast, satellite-derived indices emerge as some of the most influential predictors. Specifically, high values of FAPAR (Fraction of Absorbed Photosynthetically Active Radiation) and NDVI (Normalized Difference Vegetation Index) are generally associated with a reduced predicted fire hazard. This negative association is particularly pronounced in neighboring

spatial locations, highlighting the spatial consistency of vegetation health as a protective factor against fire risk.

Elevated mean temperatures and low FAPAR values from the previous week tend to increase the predicted fire probability. However, the model attributes a surprisingly strong positive effect to high precipitation amounts from the previous week, which traditionally would be expected to decrease fire hazard by moistening fuels. This counterintuitive finding might reflect complex interactions in the data, such as delayed effects of rainfall on vegetation growth or possible noise in the precipitation measurements.

Elevation of neighboring areas continues to show a clear inverse relationship with fire risk, consistent with known climatic gradients.

Among distance-based features, the distance to main roads exhibits the most notable influence: greater distance from roads is associated with a lower predicted fire hazard. Similarly, distance to railways maintains a comparable but weaker negative effect.

The SHAP analysis of both models reveals several consistent patterns in fire hazard prediction for the UK. Both the XGBoost and neural network models identify satellite-derived vegetation indices (FAPAR and NDVI) as key protective factors, with higher values consistently reducing predicted fire risk. Distance-based features, particularly proximity to transportation infrastructure like railways and roads, emerge as important risk factors across both models, highlighting the role of human activity in fire ignition. Temperature-related variables show complex, sometimes counterintuitive effects that may reflect the intricate temporal dynamics of fire risk. While the models differ in their treatment of certain variables—such as the neural network's unexpected positive association between previous week's precipitation and fire risk—the overall patterns align well with established fire science principles. These findings demonstrate that SHAP values provide valuable insights into model behavior while revealing the multifaceted nature of wildfire risk factors in regions experiencing changing climate conditions.

6. Conclusion

This thesis explored the game-theoretic foundation of cooperative games and Shapley values, emphasizing their desirable properties across different payoff functions. After that I provided an overview of explainable artificial intelligence methods, highlighting various approaches to interpret model predictions and demonstrating how Shapley values offer a principled framework for post-modeling, local feature importance attribution.

In particular, I examined two main estimation techniques for Shapley values: the sampling-based approach and the SHAP framework, which approximates Shapley values via weighted linear regression. Within SHAP, I focused on Kernel SHAP–a model-agnostic method–and Tree SHAP, an efficient, exact algorithm tailored for tree-based models.

Through multiple experiments implemented in R, I compared these estimation methods across simpler models such as linear regression and logistic regression, and tree-based models. The results indicate that Kernel SHAP and Tree SHAP consistently produce accurate explanations, with Tree SHAP offering a significant advantage in computational speed for tree ensembles.

As a practical application, I conducted a case study on fire hazard prediction in the UK. I presented the dataset, which I contributed to compiling, and analyzed our existing XGBoost model. Additionally, I trained a multi-layer perceptron neural network incorporating novel spatial and temporal aggregate features. Using Python, I applied Tree SHAP to interpret the XGBoost model and Kernel SHAP for the neural network. The explanation results were comparable and aligned well with domain knowledge, reinforcing the reliability and interpretability of these methods in real-world scenarios.

The primary goal of this thesis was to demonstrate the mathematical foundation of Shapley values and to examine their practical usefulness, complemented by a real-world case study. The findings confirm that Shapley value-based methods provide both theoretically sound and practically effective tools for explaining complex machine learning models.

References

[1] B. Babolcsay. Github reposatory. `https://github.com/BabolcsayBarbara/BB_thesis_materials`, 2025.

[2] C. Burton, A. Ciavarella, D. I. Kelley, A. J. Hartley, M. McCarthy, S. New, R. A. Betts, and E. Robertson. Very high fire danger in uk in 2022 at least 6 times more likely due to human-caused climate change. *Environmental Research Letters*, 20(4):044003, 2025.

[3] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of operations research*, 19(2):257–266, 1994.

[4] K. Halász. Kaliforniai erdőtűz-kockázat modellezése. `https://www.math.elte.hu/thesisupload/thesisfiles/2023bsc_alkmat3y-cj49ch.pdf`, 2023.

[5] V. Hassija, V. Chamola, A. Mahapatra, A. Singal, D. Goel, K. Huang, S. Scardapane, I. Spinelli, M. Mahmud, and A. Hussain. Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, 16(1):45–74, 2024.

[6] A. R. Karlin and Y. Peres. *Game theory, alive*, volume 101. American Mathematical Soc., 2017.

[7] S. Lundberg. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

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

[9] J. P. László Végh, Tamás Király. Játékelmélet jegyzet. `https://tkiraly.web.elte.hu/students/jatekelmelet_jegyzet.pdf`, 2024.

[10] D. Minh, H. X. Wang, Y. F. Li, and T. N. Nguyen. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, pages 1–66, 2022.

[11] R. Mitchell, J. Cooper, E. Frank, and G. Holmes. Sampling permutations for shapley value estimation. *Journal of Machine Learning Research*, 23(43):1–46, 2022.

[12] X. Ren, Z. Xing, X. Xia, D. Lo, X. Wang, and J. Grundy. Neural network-based detection of self-admitted technical debt: From performance to explainability. *ACM transactions on software engineering and methodology (TOSEM)*, 28(3):1–45, 2019.

[13] L. S. Shapley. A value for n-person games. In *Contributions to the theory of games, vol. 2*, Ann. of Math. Stud., no. 28, pages 307–317. Princeton Univ. Press, Princeton, NJ, 1953.

[14] Y. Shoham and K. Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[15] E. Strumbelj and I. Kononenko. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18, 2010.

[16] E. Štrumbelj and I. Kononenko. A general method for visualizing and explaining black-box regression models. In *Adaptive and Natural Computing Algorithms: 10th International*

Conference, ICANNGA 2011, Ljubljana, Slovenia, April 14-16, 2011, Proceedings, Part II 10, pages 21–30. Springer, 2011.

[17] E. Štrumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41:647–665, 2014.

AI Usage Statement

I, Barbara Babolcsay, declare that during the preparation of my thesis, I used the specified AI-based tools to perform the following tasks:

Task	AI tool	Place of usage
Spelling and grammar checking	Claude Sonnet 4	Entire thesis
Debugging during coding	GPT-4o	Section 5

Apart from those listed above, I did not use any other AI-based tools.