České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

COOPERATIVE GAME THEORY FOR MACHINE LEARNING TASKS

Kooperativní teorie her pro úlohy strojového učení

Master's thesis

Autor:	**Bc. Jan Pecka**
Vedoucí práce:	**doc. Ing. Tomáš Kroupa, Ph.D.**
Akademický rok:	2019/2020

České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská

Katedra: matematiky Akademický rok: 2018/2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student:	Bc. Jan Pecka
Studijní program:	Aplikace přírodních věd
Studijní obor:	Matematické inženýrství
Název práce (česky):	Kooperativní teorie her pro úlohy strojového učení
Název práce (anglicky):	Cooperative game theory for machine learning tasks

Pokyny pro vypracování:

1) Seznamte se s hlavními modely používanými v teorii koaličních her. Pozornost věnujte především axiomatické charakterizaci lineárních operátorů hodnoty pomocí eficience a symetrie (Shapleyho a Banhzafova hodnota). Prostudujte metody jejich odhadu pomocí Monte Carlo simulací a odvoďte vlastnosti použitých estimátorů.

2) Analyzujte vlastnosti koaličních her, které se používají v dostupné literatuře v oblasti strojového učení, zejména z hlediska jejich vlastností (superaditivita, supermodularita) a výpočetní náročnosti jejich reprezentace. V dalším kroku analýzu rozšiřte i na vlastnosti operátorů hodnoty běžně používaných v současné literatuře (Shapley) a zhodnoťte možnosti použití alternativních operátorů vzhledem ke splnění různých axiómů operátorů hodnoty.

3) Pomocí experimentů ve vhodném SW nástroji porovnejte výsledky klasifikačních úloh pro Shapleyho hodnotu a výsledky dosažené pro nový přístup, který je popsán výše.

Doporučená literatura:

1) E. Strumbelj and I. Kononenko, Explaining prediction models and individual predictions with feature contributions. Knowledge and information systems 41, 2014, 647–66.
2) R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. ACM computing surveys (CSUR), 51(5), 93, 2018.
3) R. J. Aumann, S. Hart, Handbook of game theory with economic applications, Volume 3 of Handbooks in Economics. North-Holland Publishing Co., Amsterdam, 2002.
4) B. Peleg and P. Sudholter. Introduction to the theory of cooperative games, volume 34 of Theory and Decision Library. Series C: Game Theory, Mathematical Programming and Operations Research. Springer, Berlin, second edition, 2007.
5) J. Castro, D. Gómez, and J. Tejada. Polynomial calculation of the Shapley value based on sampling. Computers \& Operations Research, 36(5):1726–1730, 2009.

Jméno a pracoviště vedoucího diplomové práce:

Doc. Ing. Tomáš Kroupa, Ph.D.
Centrum umělé inteligence, Fakulta elektrotechnická ČVUT, Karlovo náměstí 13, Praha

Jméno a pracoviště konzultanta:

Datum zadání diplomové práce:	28.2.2019
Datum odevzdání diplomové práce:	6.1.2020

Doba platnosti zadání je dva roky od data zadání.

Acknowledgments:
I would like express my sincere thanks to my supervisor, Tomáš Kroupa[1], for his helpful demeanor, his wealth of knowledge that has often shaped the direction of this thesis, his many ideas without which this thesis could not exist, and his careful reading of the text. Furthermore, I wish to thank Jaroslav Hlinka[2] who has provided us with a dataset and an interesting research question that is at the center of the last chapter.

[1]Artificial Intelligence Center, Faculty of Electrical Engineering, Czech Technical University in Prague

[2]Department of Nonlinear Dynamics and Complex Systems, Institute of Computer Science, Academy of Sciences of the Czech Republic

Declaration:

I hereby declare that this thesis is entirely my own work and I have cited all the used sources in the bibliography.

Prague, June 2020

Bc. Jan Pecka

Název práce:

Kooperativní teorie her pro úlohy strojového učení

Autor: Bc. Jan Pecka

Obor: Matematické inženýrství

Druh práce: Diplomová práce

Vedoucí práce: doc. Ing. Tomáš Kroupa, Ph.D., Centrum umělé inteligence, FEL ČVUT

Abstrakt: S popularizací komplikovaných metod strojového učení, tvořících komplexní a netransparentní modely bez možnosti jejich intuitivního pochopení, sílí také požadavky po vytvoření interpretačních technik, které by pomohly s vysvětlením toho, jak takové modely dospěly ke svým závěrům. Jedna z nejpoužívanějších metod pro interpretaci, a střed zájmu této práce, je postavená na Shapleyho hodnotě z koaliční teorie her. Na následujících stránkách prozkoumáme teoretické základy této hodnoty a porovnáme její výsledky k jinému konceptu teorie her, Banzhafově hodnotě, a to jak z teoretického, tak praktického hlediska. Po důkladném studiu vlastností těchto hodnot budeme schopni navrhnout nový způsob interpretace kategorických proměnných, a navíc na jednoduchém příkladu ukázat, proč může současný způsob vést k chybným výsledkům. V poslední kapitole pak detailně prozkoumáme data pocházející z měření mozkové aktivity a s pomocí Shapleyho hodnoty odhalíme zajímavá spojení mezi jednotlivými oblastmi lidského mozku.

Klíčová slova: automatic relevance determination, gaussovské procesy, klasifikace, regrese, výběr proměnných

Title:

Cooperative game theory for machine learning tasks

Author: Bc. Jan Pecka

Abstract: The recent rise in popularity of complex machine learning models trained by numerical optimization has led to an increased interest in interpretation methods capable of explaining the decisions reached by those models. One of the most common interpretation method, and the centerpiece of this thesis, is based on the Shapley value from coalitional game theory. In this text we study its theoretical foundations from a number of perspectives and compare its results, both theoretically and on an applied example, to a different solution concept, the Banzhaf value. We propose a new way of interpreting categorical variables built upon axioms of coalitional game theory and show on a counterexample why the current way leads to wrong results. Finally, we end this text with an extensive analysis of a dataset containing measurements of brain activity where we use the Shapley value to discover interesting connections among the brain regions.

Key words: Banzhaf value, Coalitional game theory, Explainable AI, Interpretation methods, Shapley value

Contents

Introduction

The rapid surge of interest in machine learning in recent years, both in academia and in the private sphere, has been accompanied by an increasing demand for interpretation techniques capable of explaining decisions of the often intricate and unintelligible models. Due to rising capabilities of contemporary processors, popularization of cloud technologies, abundance of big data, and open-source software for their processing, deep statistical models trained by advanced optimization algorithms have become commonplace in various areas ranging from retail and fintech to law and medicine. However, these models have also become more and more obscure, essentially transforming into black boxes that provide the user with a decision, but no means of comprehending the reasoning in the background.

In the last decade we have seen a push towards creating methods for interpretation of these black boxes so that we can understand them better and prevent employing faulty models in crucial or sensitive areas. The most important method for the purposes of this thesis comes from Štrumbelj and Kononenko [57], who have used coalitional game theory, specifically the concept of Shapley value, to quantify the impact each of a model's features has on the model's output. By stochastically perturbing the input values of every possible subset of features and aggregating back from subsets to individual features through the Shapley value, the authors provide intuitive graphical explanations for any particular decision made by the model. Their work has been further expanded, for instance, by Frye, Feige, and Rowat [22] and Lundberg and Lee [33], and it has become a benchmark for any other work in the area.

In this thesis we aim to further explore the connection between coalitional game theory and interpretation methods by setting a firm theoretical foundation of game theory which we can then further develop in the context of interpretable machine learning. The Shapley value, albeit perhaps the best known solution, is by no means the only one – another promising concept is the Banzhaf value. Their use in interpretation has thus far been only very scarcely explored and we wish to elaborate upon their suitability further, both from a theoretical and practical perspective.

As far as we are aware, this thesis contains several contributions and novel ideas that have not yet been included elsewhere:

- Rigorous definitions of both coalitional game theory and its use in interpretation. Through studying the axiomatic properties of various concepts from game theory, we will be able to identify their potential misuse in the way they are currently commonly applied to interpreting categorical variables, provide a simple counterexample and come up with the correct solution. More about this in Section 2.3.

- Thorough comparison of the Shapley and Banzhaf value and their use in the context of interpretation. We will study their relation both from a theoretical perspective and mainly through an extensive mathematical comparison of their results on several real world datasets and models. This will be the topic of Chapter 3.

- Extensive example of using the described interpretation methods on a dataset describing brain activity with the goal of discovering substantial connections among regions of the human brain. This will be covered in the final Chapter 4.

We will start this thesis with Chapter 1 where we shall introduce coalitional game theory in greater detail, study its most common solutions, and compare their various axiomatic definitions. We will see why their calculation requires exponential complexity and provide sampling algorithms that reduce the complexity to a polynomial level.

Chapter 2 is devoted to the connection between coalitional game theory and model interpretation. We will start with further expanding on the need for interpretation methods and delve into the vast literature that has surrounded this topic. The use of Shapley value for interpretation will be thoroughly defined and studied, and we will also adapt the sampling algorithms from the previous chapter.

The last two chapters form the practical core of this thesis where we shall apply the knowledge obtained in the preceding chapters and interpret a number of nonlinear models. In Chapter 3 we will compare the Shapley and Banzhaf value in the context of machine learning, while in Chapter 4 we will train a complicated model for time series prediction and then use the Shapley value to gain meaningful insights.

Chapter 1

Coalitional games and their values

As mentioned in the introduction, we make use of the framework of coalitional game theory for analyzing and interpreting opaque machine learning models. In this chapter we focus solely on game theory and delve deep into its foundations, concepts, and solutions without heavily exploring their connection to machine learning and interpretability. That will be the topic of the next chapter.

Game theory encompasses studies of various types of scenarios (games) where any number of rational agents (players) is tasked with a decision making process. Its goals include questions such as finding the optimal behavior of players yielding the biggest outcome, allocating cost of a shared project among a group of players, or finding the most likely outcome given each player's preferences. Many of its concepts have become widely known even outside the field itself, these include for instance zero-sum games, the prisoner's dilemma, Nash equilibrium, and others.

However, for the purpose of this thesis, we are interested in one of its branches in particular, the coalitional game theory.[1] Here players do not oppose each other, but are given a choice to enter in a coalition with any number of other players. The game then assigns a worth to every such coalition that may emerge. This naturally gives rise to several questions that coalitional game theory is tasked with answering: Which coalitions are most likely to occur? When assigned the final worth, how should it be fairly distributed among members of the winning coalition with respect to each individual's contribution? Given a typical environment where coalitions are formed, such as parliament, what is the power of each of its members?

To explore these topics further, we will first turn to fundamental definitions and theorems. We shall define games and players mathematically, see some of their properties, and present simple examples. This is the topic of the first section in this chapter.

In the next part we explore various ways of solving a game. First of all, in Section 1.2 we define what we mean by finding a solution and introduce a straightforward example called the core. Several such solutions exist, our main focus throughout this entire thesis will be on value operators. These are defined in Section 1.3 and their axiomatic nature is studied. A standard set of axioms yields the most widely used value operator, the Shapley value – this is the content of Section 1.4, while other operators, for example the Banzhaf value, are presented in Section 1.5. In Section 1.6 we explore how the Banzhaf value differs from the Shapley value through a different set of axioms studying mergers of players.

A solution is not worth much if it cannot be calculated in a reasonable amount of time. Thus, in the last section of this chapter, we discuss computational demands of the preceding solutions and describe algorithms for their approximation.

[1] COALITIONAL GAME THEORY is also often referred to as COOPERATIVE GAME THEORY.

This chapter is mainly inspired by books written by Maschler, Solan, and Zamir [34], Osborne and Rubinstein [39], and Owen [40], that provide a great starting point as well as in-depth material for every major subtopic of game theory. Other sources are cited wherever relevant.

1.1 Fundamentals of coalitional game theory

Here we lay down the mathematical groundwork and present fundamental definitions of coalitional game theory together with two elementary examples of games.

Let us start by setting up notation for players and coalitions. In this text we limit ourselves to studying finite games, i.e., games with a finite number of players.

Definition 1. Let N be a finite set such that $|N| = n \in \mathbb{N}$.[2] We call members of this set PLAYERS and every subset $S \subseteq N$ a COALITION. Furthermore, n is referred to as the GAME SIZE. Naturally, 2^N denotes the set of all possible coalitions in N.

Now we can move on to defining the game itself, which is sufficiently accomplished by specifying the resulting outcome of forming a coalition.

Definition 2. A COALITIONAL GAME is a pair (N, v) where

- N is a group of players from Definition 1.
- $v : 2^N \to \mathbb{R}$, such that $v(\emptyset) = 0$, is a COALITIONAL FUNCTION assigning to each possible coalition its WORTH.[3]

To simplify things we will refer to a coalitional game only by its coalitional function v whenever the group of players N is understood.

By G we denote the linear space of all coalitional games where for every pair of games $v, w \in G$, $\alpha \in \mathbb{R}$, and every coalition $S \subseteq N$, we define operations $(+, \cdot, \wedge, \vee)$:

$$\begin{aligned}(v + w)(S) &\equiv v(S) + w(S)\\ (\alpha \cdot v)(S) &\equiv \alpha \cdot v(S)\\ (v \wedge w)(S) &\equiv \min\{v(S), w(S)\}\\ (v \vee w)(S) &\equiv \max\{v(S), w(S)\}.\end{aligned} \tag{1.1}$$

Coalitional game theory thus arises when we want to study systems where a set of agents (players) can work together and form groups to achieve a common goal. Every such player contributes a different amount towards solving the task at hand and, importantly, players' contributions depend on other members in their group.

This means that a pair of players might be efficient in solving the problem and generate larger worth than the coalition of all players N, while another pair might not be able to work together well and generate naught. This allows for an effective capture of their correlations, calculation of each player's overall strength, and fair distribution of the final outcome among the players.

To gain a clearer understanding of the concept, we take a look at several concrete examples of coalitional games.

[2]According to this definition, an empty set or a set containing only one player are also referred to as coalitions.

[3]Note that the worth of a coalition S does not depend on the actions of players outside of S.

Example 3 (Profit game)**.** Let us imagine we are in charge of putting together a team of graduates and we are considering three candidates denoted by α, β, and γ. We have tested their abilities thoroughly and determined the resulting daily income their work would generate, both individually and while working in teams.

Both α and β have strong technical background and their work alone would be well appreciated on the market. α comes with previous work experiences and can work a bit faster, her results alone would generate €10 per day. β needs a bit of time setting in and his work could for now be sold for €8 for a day. Putting them together in a team would simply add their work together since they cannot cooperate well, resulting in €18 every day.

On the other hand, γ lacks technical capabilities altogether and focuses on effective management and motivation of other members in his team. This means that γ alone would not be able to come up with meaningful outcomes and only generate €2. Bringing γ into an already existing team, however, results in a considerable boost in productivity and faster work environment where synergies between the other members are well leveraged. γ's involvement thus causes the outcome to be not simply the sum of the worth of the other members, but comes with an additional extra bonus.

The entire coalitional game is defined as

$$\begin{aligned} v(\{\emptyset\}) &= 0 \\ v(\{\alpha\}) &= 10 \\ v(\{\beta\}) &= 8 \\ v(\{\gamma\}) &= 2 \\ v(\{\alpha,\beta\}) &= 18 \\ v(\{\alpha,\gamma\}) &= 14 \\ v(\{\beta,\gamma\}) &= 12 \\ v(\{\alpha,\beta,\gamma\}) &= 25. \end{aligned}$$

We will return to this example and solve the game in Section 1.4.

Example 4 (Simple games)**.** Simple games are a special case of coalitional functions where $v : 2^N \to \{0, 1\}$. A coalition is then described either as winning, or losing, depending on the outcome. One such example might be majority voting games, defined as

$$v(S) = \begin{cases} 0 & ,|S| < \frac{n}{2} \\ 1 & ,|S| \geq \frac{n}{2}, \end{cases} \tag{1.2}$$

whose relation to, for instance, parliament voting is apparent.

Although simple games are a special example of general coalitional games, they are not, apart from a couple of exceptions, treated differently. We will point out these exception whenever they arise.

There are two special classes of coalitional games that are worth highlighting.

Definition 5. A game $v \in G$ is called SUPERADDITIVE if

$$v(S) + v(T) \leq v(S \cup T) \tag{1.3}$$

for every two coalitions $S, T \subseteq N$ such that $S \cap T = \emptyset$.

In a superadditive game a union of any pair of disjoint coalitions receives a bigger or equal reward than its members playing individually. This serves as a basis for assuming that the grand coalition N will form since it can expect to have the largest worth. We can easily verify that the game in Example 3 is superadditive.

Definition 6. A game $v \in G$ is called monotonic if

$$v(S) \leq v(T) \tag{1.4}$$

for every two coalitions $S, T \subseteq N$ such that $S \subseteq T$.

1.2 Solving a coalitional game

Now that we have set down the fundamentals and seen a couple of examples, let us turn to defining what it means to solve a game and presenting the most common ways of doing so.

There are two elementary types of questions to ask when analyzing a coalitional game: Which coalition is the most likely to form?

How should we fairly distribute the worth of a coalition between its members? What is each player's power and impact on the game? In this text we will only be concerned with the second pair of questions as the first one is rather hard to solve and will not be relevant for the following applications. In the context of answering these questions we assume the grand coalition N forms, which is in line with the assumption of superadditivity.

Definition 7. Let $v \in G$ be a coalitional game. Any mapping $\phi : v \in G \mapsto \phi(v) \subseteq \mathbb{R}^n$ is called a SOLUTION to v.

A solution associates a coalitional game v with a set of vectors $\phi(v) \subseteq \mathbb{R}^n$. We can interpret any such vector $\boldsymbol{x} \in \phi(v)$ as a distribution of the game's worth among its n players where player i receives the amount x_i.

With this definition in mind we can further impose a couple of additional intuitive properties that constitute a well known type of a solution, the core.

Definition 8. Let $v \in G$ be a coalitional game. Its CORE $C(v)$ is defined as[4]

$$C(v) \equiv \left\{ \boldsymbol{x} \in \mathbb{R}^n ; \left(\sum_{i \in S} x_i \geq v(S) \ , \forall S \subset N \right) \wedge \left(\sum_{i=1}^{n} x_i = v(N) \right) \right\}. \tag{1.1}$$

Note that the first condition also implies that $x_i \geq v(i)$. Altogether this means that every coalition $S \subset N$ receives at least as much as is its worth, the worth of the grand coalition is fully distributed, and each player gets at least her individual worth. Such a solution makes it rational for every player to enter into coalitions as their worth does not decrease and the final worth can be distributed easily according to $\boldsymbol{x}$.

The set of inequalities defining the core might not, however, have a solution, resulting in an empty set of vectors $C(v)$. The most general theorem concerning the existence of a non-empty core comes from Bondareva [7] and Shapley [49] who have both proven it independently of each other.

[4]We can easily see that the core is an example of a solution.

Theorem 9 (Bondareva, Shapley)**.** *The core of a coalitional game v is non-empty if and only if the game satisfies*

$$v(N) \geq \sum_{S \in 2^N \setminus \{\emptyset\}} \delta(S)\, v(S) \tag{1.2}$$

for every function $\delta : 2^N \setminus \{\emptyset\} \to \mathbb{R}^+$ such that

$$\sum_{\substack{S \in 2^N \setminus \{\emptyset\} \\ i \in S}} \delta(S) = 1 \quad , \forall i \in N. \tag{1.3}$$

The core represents a set of stable solutions: if $\boldsymbol{x} \in C(v)$, then each player should be content with the results and is not expected to leave the coalition. In our latter applications, however, we are not interested in a stable state of equilibrium, but rather in a measure that uniquely distributes the worth of the game among its players. That is why we prefer to use value operators, solutions yielding a single distribution vector $\boldsymbol{x}$.

1.3 Value operators

Whereas Definition 7 allows for a general solution linking a game to a set of vectors, from now on we will only consider solutions resulting in a single vector and then impose axioms on these solutions to obtain desirable properties.

Definition 10. A VALUE OPERATOR is a mapping $\varphi : G \to \mathbb{R}^n$.

By analyzing a coalitional game $v \in G$ we obtain a vector of payoffs $\varphi(v) = (\varphi_1, \ldots, \varphi_n)$ describing each player's power or earnings from the game. This definition by itself is too general to be useful as any arbitrary vector could be picked as a solution. To tie the value operator to the coalitional function and allow for a simple interpretation of its results in the context of outcome distribution, we now introduce four standard axioms for its properties.

Definition 11 (Dummy player axiom)**.** A value operator φ satisfies the DUMMY PLAYER property if for every coalitional game $v \in G$ and any player $i \in N$ such that $v(S \cup \{i\}) = v(S)$ for every $S \subset N$,

$$\varphi_i(v) = 0. \tag{1.1}$$

This axiom ensures that a player who does not contribute anything to any coalition will receive nothing from the final share.

Definition 12 (Symmetry axiom)**.** Let $v \in G$ be a coalitional game and $i, j \in N$ a pair of players that are symmetric, i.e., $v(S \cup \{i\}) = v(S \cup \{j\})$ for every $S \subseteq N \setminus \{i, j\}$. A value operator φ is SYMMETRIC if for any such game and pair,

$$\varphi_i(v) = \varphi_j(v). \tag{1.2}$$

Two players contributing exactly the same to every coalition thus receive the same reward. Or, in another words, two players differing only by their name are entitled to the same outcome.

Definition 13 (Efficiency axiom)**.** A value operator φ is EFFICIENT if for any coalitional game $v \in G$,

$$\sum_{i=1}^{n} \varphi_i(v) = v(N). \tag{1.3}$$

If we assume that the grand coalition N will form and we wish to distribute the resulting payoff, this property makes it easy to do so and guarantees that the entire worth of the grand coalition will be assigned.

Definition 14 (Additivity axiom)**.** A value operator φ is ADDITIVE if for any pair of coalitional games $v, w \in G$ defined on the same set of players N,

$$\varphi(v + w) = \varphi(v) + \varphi(w). \tag{1.4}$$

This axiom is not applicable to simple games (see Example 4), since we cannot automatically assume that $v + w$ will be simple as well[5]. For this reason, the additivity axiom is in the context of simple games replaced by the following one defined by Dubey [18].

Definition 15 (Additivity for simple games)**.** A value operator φ is ADDITIVE for any two simple coalitional games $v, w \in G$ defined on the same set of players N, if

$$\varphi(v \wedge w) + \varphi(v \vee w) = \varphi(v) + \varphi(w). \tag{1.5}$$

These four axioms are essential to the task of finding a value operator for the sake of distributing the game's worth as each of them provides the operator with an indispensable characteristic. Quite remarkably then, they are also enough to define one and only one such operator called the Shapley value.

1.4 The Shapley value

Shapley proved in his seminal article [48] that there exists only one value operator satisfying all of the above axioms simultaneously. We will now construct this operator.

To determine the impact a player has on a game $v \in G$, we use the player's marginal contributions to arbitrary coalitions – how much his addition to a coalition changes the outcome of the game.

Definition 16. Let $v \in G$ be a coalitional game and $i \in N$ its player. i's MARGINAL CONTRIBUTION to coalition $S \subseteq N \setminus \{i\}$ is defined as

$$\Delta_i^v(S) \equiv v(S \cup \{i\}) - v(S). \tag{1.1}$$

To gain a player's overall influence on the entire game, we now only have to weigh his marginal contributions across every possible coalition. Shapley does this through calculating the expected value of the player's marginal contributions with respect to a uniform distribution across the set of all possible permutations on the set of players N. This will be elaborated upon on in the subsequent paragraphs.

Let us first denote the set of all permutations – bijective functions from a set N onto itself – as $\Pi(N)$[6]. Every permutation $\pi \in \Pi(N)$ defines an order on the set of players N and when calculating the value of player i we can imagine that this order determines the formation of a coalition: every player preceding i in the permutation has already joined the coalition, whereas subsequent players will only join later. We can calculate the marginal contributions of i's arrival for every such permutation and, since there is no preference of one permutation over another, weigh them with uniform weights.

To put this into mathematical terms, we will denote by $\mathrm{Pre}^i(\pi)$ the set of players preceding player i in a permutation π:

$$\mathrm{Pre}^i(\pi) = \{j \in N; \pi(j) < \pi(i)\}. \tag{1.2}$$

Note that $i \notin \mathrm{Pre}^i(\pi)$. Now we have everything prepared for the definition of the Shapley value.

[5]Refer back to for definitions of the $(+, -, \wedge, \vee)$ operators.

[6]Naturally there are $n!$ possible permutations over N.

Definition 17. Let $v \in G$ be a coalitional game and $i \in N$ its player. The Shapley value of player i is the value operator with the following definition:

$$\begin{aligned}\varphi_i^{\mathrm{Sh}}(v) &\equiv \frac{1}{n!}\sum_{\pi\in\Pi(N)} \Delta_i^v\left(\mathrm{Pre}^i(\pi)\right) \qquad (1.3)\\ &\overset{(1.1)}{=} \frac{1}{n!}\sum_{\pi\in\Pi(N)} \left[v\left(\mathrm{Pre}^i(\pi)\cup\{i\}\right) - v\left(\mathrm{Pre}^i(\pi)\right)\right].\end{aligned}$$

Theorem 18 (Shapley [48])**.** *The Shapley value is the unique value operator simultaneously satisfying the dummy player property, symmetry, efficiency, and additivity.*

We can also think of Equation (1.3) as an expected value of marginal contributions,

$$\varphi_i^{\mathrm{Sh}}(v) = \mathbf{E}_{\mathbf{P}}\left[\Delta_i^v\left(\mathrm{Pre}^i(\pi)\right)\right], \qquad (1.4)$$

where P is a uniform probability distribution over $\Pi(N)$. Later we will encounter value operators with different probability distributions.

There is a second equivalent way of defining the Shapley value that is often useful, this time using marginal contributions of player i to all the explicitly given coalitions excluding the player, meaning

$$\varphi_i^{\mathrm{Sh}}(v) = \sum_{S\subseteq N\setminus\{i\}} w_S^{\mathrm{Sh}}\Delta_i^v(S). \qquad (1.5)$$

To calculate the weights w_S^{Sh} from Equation (1.3), let's imagine a coalition $S \subseteq N\setminus\{i\}$ and count the number of permutations whose order results in players in S being the predecessor of i. In other words,

$$w_S^{\mathrm{Sh}} = \frac{1}{n!}\left|\left\{\pi; S = \mathrm{Pre}^i(\pi), i \in N\right\}\right|. \qquad (1.6)$$

There are $|S|!$ ways to order players in S ahead of i and $(n-|S|-1)!$ ways to order the remaining players behind i. This, multiplied by $1/n!$ gives the following equivalent definition of the Shapley value.

Remark 19. The Shapley value can be equivalently expressed as[7]

$$\begin{aligned}\varphi_i^{\mathrm{Sh}}(v) &= \sum_{S\subseteq N\setminus\{i\}} \frac{|S|!\,(n-|S|-1)!}{n!}\Delta_i^v(S)\\ &= \sum_{S\subseteq N\setminus\{i\}} \frac{1}{n\binom{n-1}{|S|}}\Delta_i^v(S). \qquad (1.7)\end{aligned}$$

Example 20. In Example 3 we've presented a profit game with players α,β,γ and studied their contributions toward a team's efficiency. Remember, that α and β are useful team members, both generating valuable results, but they lack the ability to cooperate. On the other hand, γ's abilities are best utilized when put in a group where he can boost the productivity of other team members.

Straightforward calculations give us their Shapley values as

$$\begin{aligned}\varphi_\alpha^{\mathrm{Sh}} &= 11.3\\ \varphi_\beta^{\mathrm{Sh}} &= 9.3\\ \varphi_\gamma^{\mathrm{Sh}} &= 4.3.\end{aligned}$$

[7]If we were to randomly draw coalitions $S \subseteq N\setminus\{i\}$ for calculation of the Shapley value, we would first select the coalition's cardinality $|S|$ with equal probabilities and then select one of the $\binom{n-1}{|S|}$ coalitions with that cardinality.

Notice, that the values sum up to the value of the grand coalition (25) and that they correctly capture the added efficiency resulting from cooperation, ascribing players with larger values than are their individual worths (defined to be 10, 8, and 2 for players α,β, and γ respectively). Player γ receives the biggest boost, as he is the one encouraging cooperation while not producing much value individually.

Example 21. For simple games, the Shapley value is more commonly referred to as the Shapley–Shubik power index [50]. We say that a player i is PIVOTAL to a coalition $S \subseteq N$ whenever $\Delta_i^v(S) = 1$, or, in another words, whenever her arrival to a losing coalition turns it into a winning one. The player's Shapley value is then equal to the number of times this player is pivotal, with coalitions weighted with the same weights as in eq: Shapley value equiv:

$$\varphi_i^{\text{S–S}}(v) \equiv \sum_{\substack{S \in N\setminus\{i\} \\ i \text{ is pivotal}}} \frac{1}{n\binom{n-1}{|S|}}. \tag{1.8}$$

This index has been repeatedly used for analysis of voting power in the Council of the European Union, see for instance Barr and Passarelli [5] and Varela and Prado-Domínguez [60]. It has been modified by Owen and Shapley [41] to account for player's individual preferences toward joining coalitions.

A big advantage of using value operators for solving coalitional games is their axiomatic nature. Here we have (for now) only introduced the standard set of axioms defining the Shapley value, but even these could be changed to a different set, seemingly unrelated, and produce the same operator. For more axioms concerning the Shapley value and other uniqueness theorems, see Winter [64] or Section 1.6 where we present several instances of other axioms.

A note on terminology Whenever we use the term SHAPLEY VALUE without highlighting a concrete player, we refer to the operator φ^{Sh} from Definition 17, i.e., a mapping from G to $\mathbb{R}^n$. The actual outcomes of this operator ascribing each player with a single value – $\left(\varphi_i^{\text{Sh}}\right)_{i=1}^n$ – are referred to as their SHAPLEY VALUES. A single player's outcome φ_i^{Sh} is also referred to as his Shapley value, but the player has to be mentioned. The same distinction holds for the Banzhaf value(s) which we shall introduce later.

1.5 Quasivalues and semivalues

The Shapley value is the unique value operator satisfying all of the axioms listed above, but in many cases we might want to relax or change some of them. In this section we stick to using marginal contributions as a measure of a player's impact and study what happens when one of the axioms is not met. More detail on this topic can be found in Monderer and Samet [35].

Definition 22. Let φ be a value operator and $i \in N$ a player of a game $v \in G$. φ is called a PROBABILISTIC VALUE if there exists a probability distribution function $p^i : 2^{N\setminus\{i\}} \to [0,1], \sum_{S \subseteq 2^{N\setminus\{i\}}} p^i(S) = 1$ such that φ_i can be written as

$$\varphi_i(v) = \mathrm{E}_{p^i}\left[\Delta_i^v\right] = \sum_{S \subseteq N\setminus\{i\}} p^i(S)\,\Delta_i^v(S) \tag{1.1}$$

and $\varphi(v) = (\varphi_i(v))_{i\in N}$.

It can be shown that a probabilistic value automatically satisfies the dummy player and additivity axioms, the Shapley value is thus the unique probabilistic value that is both efficient and symmetric. We will call efficient probability values QUASIVALUES and symmetric probability values SEMIVALUES.

When we first introduced the Shapley value in Definition 17, it wasn't in the form of a probabilistic value, but rather as an expected value of marginal contributions with respect to a probability distribution over the set of permutations of N – this is called a RANDOM-ORDER VALUE. Weber [61] actually proved that a value operator is a quasivalue if and only if it is a random-order value. In our specific case we have used a uniform distribution, drawing every permutation with probability $1/n!$.

In much the same way, a value operator is a semivalue if and only if it is a probabilistic value whose probability density function only depends on the coalition size,

$$p^i(S) = p^j(T), \quad \forall i, j \in N, \quad \forall S \subseteq N\setminus\{i\}, \forall T \subseteq N\setminus\{j\}, |S| = |T|. \tag{1.2}$$

Every player then receives equal treatment and the resulting operator is symmetric.

We know that the Shapley value is defined with probabilities

$$p(S) = \frac{1}{n\binom{n-1}{|S|}}. \tag{1.3}$$

Now is a good time to ask why we don't use a uniform distribution weighing every coalition the same. Seeing that there are

$$\sum_{s=0}^{n-1}\binom{n-1}{s} = 2^{n-1} \tag{1.4}$$

possible coalitions in $N\setminus\{i\}$,we arrive at the definition of the Banzhaf value [4].

Definition 23. BANZHAF VALUE is a probabilistic value defined as

$$\varphi_i^{\mathrm{B}}(v) \equiv \sum_{S\subseteq N\setminus\{i\}} \frac{1}{2^{n-1}}\Delta_i^v(S). \tag{1.5}$$

The coalitions' probabilities of the Banzhaf value are not player-dependent, hence it is a semivalue. Since it doesn't satisfy the efficiency axiom, it is not applicable in scenarios where one wishes to distribute a game's outcome among the players, but can be used for determining a player's power in a game. We will study this distinction in latter chapters.

1.6 Alternative axiomatizations

In Section 1.3 we have presented the standard set of axioms that uniquely determine the Shapley value. These are however not the only axioms useful for analyzing properties of a value function, we will now study a different set that better describes the Banzhaf value.

Lehrer [31] studied the effect of merging players and the impact it might have on their values. He wanted to know whether merging two players into one is profitable and how this is reflected by the Shapley and Banzhaf value. To follow his steps, we first have to define a new game arising from the aforementioned unification of players.

Definition 24. Let (N, v) be a coalitional game and $T \subseteq N$ a group of players. In the AMALGAMATED T-GAME, players from T are grouped into a new player denoted by $\mathring{T}$, and the game is defined as

$$\left(N\setminus T \cup \left\{\mathring{T}\right\}, v^T\right), \tag{1.1}$$

where for each $S \subseteq N\setminus T \cup \left\{\mathring{T}\right\}$

$$v^T(S) \equiv \begin{cases} v(S) & ,\mathring{T} \notin S \\ v\left(S\setminus\left\{\mathring{T}\right\} \cup T\right) & ,\mathring{T} \in S. \end{cases} \tag{1.2}$$

Thus, players in T only enter coalitions together through the new amalgamated player $\mathring{T}$. We can then inspect the value of player $\mathring{T}$ in the new T-game and study how it relates to values of the original individual players.

We now limit ourselves to amalgamating pairs of players, i.e., $|T| = 2$, and introduce a new axiom for value operators.

Definition 25 (Superadditivity axiom)**.** A value operator φ is SUPERADDITIVE if for any coalitional game $v \in G$ and a pair of its players $\{i, j\} = T$,

$$\varphi_i(v) + \varphi_j(v) \leq \varphi_{\mathring{T}}\left(v^T\right). \tag{1.3}$$

This means that a superadditive operator ascribes bigger value to the merged pair than is the sum of its parts and makes merging profitable[8]. Lehrer than goes on to prove the following theorem that states that the Banzhaf value is the only operator satisfying this axiom and the ones introduced before.

Theorem 26 (Lehrer [31])**.** *The Banzhaf value is the unique value operator simultaneously satisfying the dummy player property, symmetry, additivity, and superadditivity.*

Casajus [9] studied superadditivity further and found redundancy in the set of axioms of Theorem 26. He proposed yet another axiom, replacing the inequality of the superadditivity axiom with an equality, creating a restricted version of the efficiency axiom for groups of two players.

Definition 27 (2-efficiency axiom)**.** A value operator φ satisfies the 2-EFFICIENCY axiom if for every coalitional game $v \in G$ and a pair of its players $\{i, j\} = T$,

$$\varphi_i(v) + \varphi_j(v) = \varphi_{\mathring{T}}\left(v^T\right). \tag{1.4}$$

He further shows that the additivity and superadditivity axioms automatically imply 2-efficiency, and this in turn implies symmetry. Theorem 26 can thus be restated only with the dummy player axiom, additivity, and superadditivity.

Moreover, 2-efficiency in itself is enough to characterize the Banzhaf value and distinguish it from others:

Theorem 28 (Casajus)**.** *The Banzhaf value is the unique value operator that satisfies both the dummy player property and 2-efficiency.*

Interestingly enough, we have arrived at a property that discriminates between the Shapley and Banzhaf values: whereas the Shapley value is efficient when all players are amalgamated into one and their values are summed up,

$$\sum_{i \in N} \varphi_i^{\mathrm{Sh}}(v) = \varphi_N^{\mathrm{Sh}}\left(v^N\right) = v(N), \tag{1.5}$$

the Banzhaf value is efficient for every pair of players,

$$\sum_{i \in T} \varphi_i^{\mathrm{B}}(v) = \varphi_{\mathring{T}}^{\mathrm{B}}\left(v^T\right), \quad \forall T \subseteq N, |T| = 2. \tag{1.6}$$

[8]This is especially useful for superadditive games. The interpretation of this axiom is less clear for other types of games, or games with both positive and negatives values.

Algorithm 1.1 Approximate sampling of a random-order value

```
INPUT: coalitional game $v$; # of samples $m \in \mathbb{N}$; probability distribution $p$ over $\Pi(N)$

OUTPUT: estimates of players' values $(\hat{\varphi}_1(v), \ldots, \hat{\varphi}_N(v))$

$\hat{\varphi}_i(v) \leftarrow 0 \quad \forall i \in N$
for sample in $m$:
    randomly draw $\pi$ from $\Pi(N)$ according to $p$
    for player $i$ in $N$:
        determine $\mathrm{Pre}^i(\pi)$
        calculate $\Delta_i^v\left(\mathrm{Pre}^i(\pi)\right) = v\left(\mathrm{Pre}^i(\pi) \cup \{i\}\right) - v\left(\mathrm{Pre}^i(\pi)\right)$
        $\hat{\varphi}_i(v) \leftarrow \hat{\varphi}_i(v) + \Delta_i^v\left(\mathrm{Pre}^i(\pi)\right)$
    end
end
$\hat{\varphi}_i(v) \leftarrow \frac{\hat{\varphi}_i(v)}{m} \quad \forall i \in N$
```

1.7 Calculation via sampling

Calculating marginal contributions for every possible coalition in N soon proves to be impossible as their number grows exponentially and these calculations would have to be carried out for each player individually. It is therefore advantageous to resort to approximate sampling techniques.

Castro, Gómez, and Tejada [10] proposed two unbiased sampling algorithms for approximating the Shapley value and a general semivalue. These operate in polynomial time and their speed hinges on the complexity of computing values of the coalitional function v.

We will slightly improve their algorithm for sampling of the Shapley value to extend its functionality to any random-order value φ defined as

$$\varphi_i(v) = \sum_{\pi \in \Pi(N)} p(\pi)\, \Delta_i^v\left(\mathrm{Pre}^i(\pi)\right), \tag{1.1}$$

where $p : \Pi(N) \to [0,1]$ is a probability distribution function over $\Pi(N)$. In each step of the sampling procedure we randomly draw a permutation π from $\Pi(N)$ and for each player calculate her marginal contribution to $\mathrm{Pre}^i(\pi)$. The sample mean of the calculated contributions gives us the desired approximation. The algorithm is summarized in pseudocode in Algorithm 1.1.

This is a simple procedure for approximating the sample mean and thus, due to the central limit theorem, several of its properties are evident: it yields estimators that, for each $i \in N$, tend to the normal distribution, are unbiased ($\mathrm{E}[\hat{\varphi}_i(v)] = \varphi_i(v)$), consistent in probability,

$$\lim_{m \to +\infty} \mathrm{P}\left(|\hat{\varphi}_i(v) - \varphi_i(v)| > \epsilon\right) = 0, \quad \forall \epsilon > 0, \tag{1.2}$$

and their variance is given by σ_i^2/m, where σ_i^2 is the variance of the original population. The referenced article also shows that the estimates are efficient (in terms of ax: efficiency) as well as the random-order value,

$$\sum_{i=1}^{n} \hat{\varphi}_i(v) = v(N). \tag{1.3}$$

The algorithm for approximating semivalues is a bit more complex, as the set of coalitions a player

Algorithm 1.2 Approximate sampling of a semivalue

```
INPUT: coalitional game $v$; player $i$, # of samples $m \in \mathbb{N}$; probability density function $p$ over $2^{N\setminus\{i\}}$

OUTPUT: estimate of player $i$'s value $\hat{\varphi}_i(v)$

$\hat{\varphi}_i(v) \leftarrow 0$
for sample in $m$:
    randomly draw $S$ from $N\setminus\{i\}$ according to $p$
    calculate $\Delta_i^v(S) = v(S \cup \{i\}) - v(S)$
    $\hat{\varphi}_i(v) \leftarrow \hat{\varphi}_i(v) + \Delta_i^v(S)$
end
$\hat{\varphi}_i(v) \leftarrow \frac{\hat{\varphi}_i(v)}{m}$
```

can enter is different from those of every other player (whereas before we used permutations, which create these coalitions using the preceding players). This means that we have to use n independent sampling procedures, one for every player. This however proves not to be a big issue, as the calculation of $v(S)$ in the latter examples is more computationally demanding than the sampling procedure itself.

Algorithm 1.2 shows the pseudocode for approximating a semivalue

$$\varphi_i(v) = \sum_{S\subseteq N\setminus\{i\}} p(S)\,\Delta_i^v(S)\,. \tag{1.4}$$

The resulting estimates are, again, unbiased and consistent in probability.

Chapter 2

Interpreting nontransparent models

In recent years we have witnessed a surge of interest in machine learning and its applicability in industry. In their annual 2019 AI Index Report[1], Stanford Institute for Human-Centered Artificial Intelligence has calculated that "between 1998 and 2018, the volume of peer-reviewed AI papers has grown by more than 300%, accounting for 3% of peer-reviewed journal publications and 9% of published conference papers", while the global machine learning industry was estimated by Zion Market Research[2] at US$ 1.58 billion in 2017 and expected to reach US$ 20.83 billion by 2024.

Models' performance has risen as well in the past decade, for instance, the accuracy of the best model in the famous annual image classification challenge ILSVRC [47] increased from 71.8% to 97.3% between 2010 and 2017.

However, accuracy has often come hand in hand with model complexity arising from nonlinear models trained via numerical optimization. The associated lack of interpretability and intuitive understanding of the model's underlying functionality has produced strong calls for development of interpretation methods capable of extracting comprehensible rules that the model follows.

We are warned against ascribing omnipotent power to black-box algorithms and uncritically accepting their decisions without further investigation [21]. Nonlinear models might erroneously discover spurious correlations and use them as a basis for prediction or amplify historical biases implicitly contained in the data [38]. On the other hand, being able to understand the model is advantageous in most scenarios and crucial in several high-risk areas such as medicine, self-driving cars, finance, justice, or science.

By understanding the model's performance, we are able to increase trust and justify its deployment and use in production. Discovering errors in its reasoning can lead to corrections and development of a better model that produces results we understand and better achieves its goals.

Furthermore, in May of 2018, the European Union General Data Protection Regulation (GDPR) took effect, whose Recital 71[3] states that "[every data subject has the right to] obtain an explanation of the decision reached" when the subject's data are used in an automated decision making process and, moreover, "[the controller should] prevent ... discriminatory effects on natural persons on the basis of racial or ethnic origin, political opinion, religion or beliefs, trade union membership, genetic or health status or sexual orientation." This has created an even bigger demand for interpretation techniques meeting these goals.

This chapter is organized as follows: we start in Section 2.1 with a short review of current interpretability methods together with their systematization. One such method uses the framework of coali-

[1] Available at https://tinyurl.com/ai-index19.

[2] The report and its summary can be found at https://tinyurl.com/zion-report.

[3] Available at https://tinyurl.com/gdpr-explain.

tional game theory developed in the previous chapter and we inspect this method in detail in Section 2.2. In Section 2.3 we propose a new way of interpreting categorical variables built upon the axioms presented above. Exact calculation of results is once again computationally infeasible and we must resort to Monte Carlo sampling: we describe the algorithms and their properties in Section 2.4. Finally, Section 2.5 is devoted to recent articles that have extended the capabilities of this method and that further analyzed its results.

2.1 Interpretation methods in current literature

The field of Explainable AI is still highly fractured, without a clear systematization or rigorous definition of goals and metrics for comparison of different approaches. In recent years we have seen a couple of surveys collecting various methods and attempts to set a firm foundation for the field. These include Doshi-Velez and Kim [15], Došilović, Brčić, and Hlupić [14], Guidotti et al. [25], and Murdoch et al. [36] – all of them comprehensive reviews summarizing the progress in this area.

There are several models generally considered interpretable by their very construction, for instance, linear or logistic regression, decision trees, or rule-based models. The rest of them need additional analysis to obtain intuitively understandable explanations. Most of the explanation techniques work post hoc, i. e., they use an already trained model and test its performance in various ways to reveal the underlying mechanisms. There are several possible outcomes of these methods, ranging from feature metrics, where each feature receives a score describing its impact on the model, to bags of rules or saliency mask for image recognition. Kopp, Pevný, and Holeňa [30] have created anomaly detectors which return a random forest explaining how the anomaly differs from majority.

The interpretation methods can be divided into two groups. GLOBAL methods provide explanations for the entire model, usually in the form of an approximation that is both understandable and faithful to the original model. Our main focus, however, will be on LOCAL methods. These take a model and explain its decisions for a single specific datapoint, without considering the behaviour on other datapoints.

Another line of division is between methods that are MODEL-SPECIFIC and those that are MODEL-AGNOSTIC. Model-specific methods are, as the name suggests, tuned to work with a single class of models and use their specifics. A prominent model-specific interpretation technique is DeepLift by Shrikumar, Greenside, and Kundaje [51], which analyzes neural network through perturbing the input values and comparing the outputs to a reference value. Model-agnostic methods, on the other hand, treat the analyzed model as a black-box and are thus capable of interpreting an arbitrary model.

Two interpretation methods in particular have received wide attention: LIME and Shap. We will study Shap in great detail over the following sections, it is, however, important to include LIME in the discussion as well, due to their connection explored later in Section 2.5.

LIME, first described by Ribeiro, Singh, and Guestrin [45], creates locally accurate understandable approximations to the underlying model. If $\mathcal{L}(\mathfrak{f}, g, \pi_x)$ denotes a distance measure between functions $\mathfrak{f}$ and g in the area defined by π_x, where $\mathfrak{f}$ is the function we wish to approximate and g is the approximation, $\Omega(g)$ denotes a measure of complexity (or interpretability) of function g, LIME then searches for an approximation that minimizes both its distance from $\mathfrak{f}$ and its complexity:

$$\xi(x) = \arg\min_{g} \left(\mathcal{L}(\mathfrak{f}, g, \pi_x) + \Omega(g)\right) \tag{2.1}$$

If we, for instance, choose for g to be a hyperplane in the feature space, we obtain a linear approximation to the model $\mathfrak{f}$ which is locally accurate and intuitively understandable thanks to its connection to linear regression. LIME has been widely used due to its simple nature and model-agnosticism, the authors have provided an open source library for Python at https://github.com/marcotcr/lime. Guo et al. [26] have further developed LIME, making it suitable for security applications.

Our focus for the rest of this chapter will be on a different local and model-agnostic method. We will employ foundations of cooperative game theory described in the previous chapter to obtain interpretations through feature importances.

Before we move on to game theory; however, it is important to acknowledge that interpretation methods have also been met with criticism. Most notably, Rudin [46] argues that by incorporating models for interpretation, we create another layer of complexity that makes it harder to intuitively understand our data. She asks whether a black-box opened by an inaccurate interpretation method counts as valid under the current legislation and presents several compelling arguments for using easily interpretable models whenever possible.

2.2 Using Shapley values for model interpretation

In their article, Štrumbelj and Kononenko [57] propose an interpretation technique that is of special interest for us in this text. It works, to put it shortly, by defining a coalitional game over a set of features of a model and then perturbing their values, hence calculating the Shapley value for each of the features. Here we analyze this technique in detail and provide algorithms for its calculation.

2.2.1 Definition of the coalitional game

The authors have originally used a different approach to arrive at this concept, where they implicitly define interactions among a subset of features, split them between the members of this subset and then they prove that this leads to the formulation of the Shapley value. Here we have decided to take a more straightforward road and define the coalitional function and its Shapley value explicitly instead.

Before we do that, however, we have to set up basic notation and concepts used throughout this chapter. We consider a feature space comprising of n random variables (features) $X_1, \ldots, X_n$ whose sample spaces are $\mathcal{X}_1, \ldots, \mathcal{X}_n$, respectively. As we did in the previous chapter, we use $N = \{1, \ldots, n\}$ to denote the set of indices of all features. Y is the target variable with sample space $\mathcal{Y}$, for now assumed to be one-dimensional $\mathcal{Y} \subseteq \mathbb{R}$. We deal with the extension to multivariate target spaces later.

We further define $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, and use P_Z to denote a joint probability distribution of a random vector $Z = (X_1, \ldots, X_n, Y)$. A dataset $\mathcal{D}$ is then the set of $m \in \mathbb{N}$ random samples from Z, each drawn independently according to P_Z:

$$\mathcal{D} = \left(\left(\boldsymbol{x}^i, y^i\right) \overset{\text{iid}}{\sim} \mathrm{P}_Z\left(\boldsymbol{x}, y\right)\right)_{i=1}^{m} \tag{2.1}$$

As Fokoué puts it in [19]: "one of the most pervading goals in both theoretical and applied statistical machine learning is to find the function $\mathfrak{f}^* : \mathcal{X} \to \mathcal{Y}$ that best captures the dependencies between the $\boldsymbol{x}^i$'s and the y^i's in such a way that, given a new random (unseen) observation $z^{\text{new}} = (\boldsymbol{x}^{\text{new}}, y^{\text{new}}) \sim \mathrm{P}_Z(\boldsymbol{x}, y)$ with $z^{\text{new}} \notin \mathcal{D}$, the image $\mathfrak{f}^*(\boldsymbol{x}^{\text{new}})$ provides a prediction of y^{new} that is as accurate and precise as possible, in the sense of yielding the smallest possible discrepancy between y^{new} and $\mathfrak{f}^*(\boldsymbol{x}^{\text{new}})$."

Finding this function is beyond the scope of this text. We refer the reader to a plethora of excellent textbooks covering the topic, such as those by Bishop [6], Goodfellow, Bengio, and Courville [24], and Murphy [37]. We assume that this function has already been found and trained on the dataset and from now on we denote it simply by $\mathfrak{f}$. We will often refer to this function as the model.

We wish to analyze this function through perturbing some of the features' values and observing the changes in its output. In order to do this, we define an extension of the standard indexing of vectors, where for each set of indices $S \subseteq N$, we consider the following vector and subspace, respectively:

$$\begin{aligned} X_S &\equiv (X_i)_{i \in S} \,, \\ \mathcal{X}_S &\equiv (\mathcal{X}_i)_{i \in S} \,, \end{aligned}$$

and $\boldsymbol{x}_S = (x_i)_{i \in S}$ for any $\boldsymbol{x} \in \mathcal{X}$. Let $\mathrm{P}_S \equiv \mathrm{P}_{X_S}$ denote the marginal probability distribution of features in S.

Finally, we denote the complement of $S \subseteq N$ relative to N with S^{c}

$$S^{\mathsf{c}} \equiv N \backslash S \tag{2.2}$$

and we use $\mathfrak{f}_{\boldsymbol{x}_S}$ to denote a section of $\mathfrak{f}$ through $\boldsymbol{x}_S$, i.e., $\mathfrak{f}_{\boldsymbol{x}_S} : \mathcal{X}_{S^{\mathsf{c}}} \to \mathcal{Y}$ such that

$$\mathfrak{f}_{\boldsymbol{x}_S}\left(\boldsymbol{x}_{S^{\mathsf{c}}}\right) \equiv \mathfrak{f}\left(\boldsymbol{x}_S, \boldsymbol{x}_{S^{\mathsf{c}}}\right). \tag{2.3}$$

This allows us to randomly sample features in S^{c} while keeping others fixed to the values $\boldsymbol{x}_S$.

Everything is now ready for the definition of the coalitional game, we do so for an arbitrary datapoint $\boldsymbol{x} \in \mathcal{X}$ in the following way. A coalitional game is defined by

1. a set of its players N, and
2. a coalitional function $v : 2^N \to \mathbb{R}$.

Here, since we define a different coalitional game for every $\boldsymbol{x}$, we specify so in the notation and use $v^{\boldsymbol{x}}$.

We consider N to be the set of the model's features and for each coalition $S \subseteq N$ we wish to calculate the change in $\mathfrak{f}$'s output when features in S are known and equal to their values in $\boldsymbol{x}_S$ compared to the situation when they are not. Ideally we would not reveal the values of $\boldsymbol{x}_{S^{\mathsf{c}}}$ to the model at all, but that is not possible for most of the available machine learning methods, we thus have to use the data distribution to calculate the expected value of the change.

The coalitional function for interpreting model $\mathfrak{f}$ at datapoint $\boldsymbol{x}$ is defined for any coalition $S \subseteq N$ as

$$v^{\boldsymbol{x}}(S) \equiv \mathbf{E}_{\mathbf{P}_{S^{\mathsf{c}}}}\left[\, \mathfrak{f}_{\boldsymbol{x}_S}\left(X_{S^{\mathsf{c}}}\right) \right] - \mathbf{E}_{\mathbf{P}_N}\left[\, \mathfrak{f}\left(X_N\right) \right]. \tag{2.4}$$

In the first expected value we fix features in S to be equal to their real values in $\boldsymbol{x}$ and consider values of features in S^{c} to be unknown, following the marginal probability distribution. We then subtract the expected value when none of the features are known, this difference quantifies the impact of knowing the values of features in S.

Marginal contributions for this game can now be calculated from Equation (1.1) as

$$\Delta_i^{\boldsymbol{x}}(S) = v^{\boldsymbol{x}}(S \cup \{i\}) - v^{\boldsymbol{x}}(S), \tag{2.5}$$

describing the change in the output obtained by disregarding feature i's value. The resulting coalitional game's Shapley value is defined, according to Equation (1.3), by

$$\varphi_i^{\mathrm{Sh}}\left(v^{\boldsymbol{x}}\right) = \sum_{S \subseteq N \backslash \{i\}} \frac{1}{n\binom{n-1}{|S|}} \Delta_i^{\boldsymbol{x}}(S). \tag{2.6}$$

We will refer to this value in the context of explaining a model $\mathfrak{f}$ as the Shapley interpreter.

In the above, we have made an important assumption that the target space is one-dimensional. When $\mathfrak{f}$ is a multivariate model, we define a different coalitional game for every dimension in $\mathcal{Y}$ and calculate each feature's contribution toward the output in this dimension independently.

2.2.2 The game's properties

This interpretation method is, if we follow the taxonomy laid out in Section 2.1, model-agnostic and local. The underlying model needs to be trained only once at the beginning of the process, the subsequent analysis uses this model only as a black-box for prediction at arbitrary datapoints. The calculations can easily be parallelized, because Shapley values for different features can be calculated independently of each other.

Notice that this concept takes into account all possible subsets of the feature set, thus perturbing multiple features at once. This results in exponential complexity; however, perturbing only one feature at a time, although it might seem intuitive, does not consider interactions between features and leads to faulty results. So, in an example in [56], the authors show how this easier approach gives values equal to 0 for both features when considering the OR model $\mathfrak{f}_{\mathrm{OR}}(x_1, x_2) = x_1 \vee x_2$ and interpreting it for datapoint $\boldsymbol{x} = (1, 1)$ when the data is uniformly distributed, i.e. each feature is equal to 1 with probability $1/2$.

We see that changing one feature at a time does not change the outcome of $\mathfrak{f}_{\mathrm{OR}}$ and would ascribe zero importance to features x_1, x_2, albeit both are clearly significant for the model's output of 1. On the other hand, using the Shapley interpreter and perturbing features in every possible coalition yields values equal to $1/8$ for both features, capturing their contributions better. The unfortunate increase in computational complexity will be dealt with in the next section of this chapter.

The theoretical groundwork for coalitional games from previous chapter gives us three important properties of the Shapley interpreter:[4]

1. It satisfies the dummy player property, i.e., a feature irrelevant to the model is assigned zero value.
2. It is symmetric, hence two different features contributing exactly the same are assigned equal values.
3. It is efficient, the sum of players' values is equal to the worth of the grand coalition N.

Specifically, efficiency tells us that

$$\sum_{i=1}^{n} \varphi_i^{\mathrm{Sh}}(v^{\boldsymbol{x}}) \;=\; v^{\boldsymbol{x}}(N) \;\overset{(2.4)}{=}\; \mathfrak{f}(\boldsymbol{x}) - \mathbf{E}_{\mathbf{P}_N}[\,\mathfrak{f}(X_N)\,]$$

and thus each feature's Shapley value describes its portion in the offset between the model's actual prediction and the prior prediction averaged across the whole feature space.

We can inspect this visually in Figure 2.1 on page 20, an explanation drawn from the SHAP Python package[5] created by Lundberg and Lee [33]. This figure interprets a support vector machine trained on the Iris flower data set and explains how each feature contributes to the difference between the model's average (base) value and the outcome class prediction for a particular flower and its characteristics. For this instance, each of the four features' values lower the model's probability estimate that this instance belongs to a specific species of Iris, *setosa*, resulting in the output probability of 0.01, with petal length being the most decisive factor.

Another way of understanding the results of the Shapley interpreter is through its relation to additive linear models of the form

$$g(\boldsymbol{x}) = \sum_{i=1}^{n} g_i(x_i) + \beta_0, \tag{2.7}$$

[4]See Axiom 11, Axiom 12, and Axiom 13.

[5]Available at https://github.com/slundberg/shap.

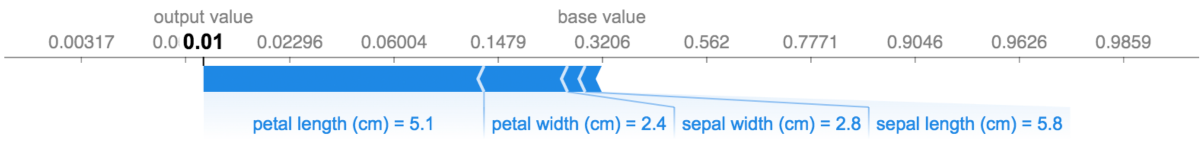


Figure 2.1: Example of local model explanation created by SHAP.

where $g_i : \mathcal{X}_i \to \mathbb{R}$ and $\beta_0 \in \mathbb{R}$ is a bias term. If we define β_0 to be equal to $\mathbf{E}_{\mathbf{P}_N}\left[\, \mathfrak{f}\,(X_N)\,\right]$ and $g_i\,(x_i) = \varphi_i^{\mathrm{Sh}}\,(v^{\boldsymbol{x}})$ for every $i \in N$, we have transformed $\mathfrak{f}$ into its additive form that is locally accurate, i.e.

$$\mathfrak{f}\,(\boldsymbol{x}) = g\,(\boldsymbol{x})\,. \tag{2.8}$$

This means that we locally approximate $\mathfrak{f}$ with a linear regression model which is easily understandable and explainable. If we do this for every $\boldsymbol{x}$, we can then calculate the output of any arbitrarily chosen model by summing up $n + 1$ terms. This is exploited in [58] where the authors create so-called nomograms, essentially reducing the calculations of a multilayered perceptron into a diagram that fits onto a piece of paper.

A largely unanswered question is the aggregation of local Shapley values for every data point into a global interpretation framework. In their original article, Štrumbelj and Kononenko use the standard deviation of $\left\{\varphi_i^{\mathrm{Sh}}\,(v^{\boldsymbol{x}})\,;\boldsymbol{x} \in \mathcal{X}\right\}$ as a proxy of feature i's importance on the model. Another, perhaps more straightforward, definition of the global value is

$$\mathbf{E}_{\mathbf{P}_N}\left[\varphi_i^{\mathrm{Sh}}\left(v^{X_N}\right)\right]. \tag{2.9}$$

This value better considers the distribution of data and could help investigate the feature's overall impact on the population. A thorough analysis and comparison of global Shapley value aggregation is, to the best of our knowledge, missing in the literature.

2.3 Interpreting dummy-encoded categorical variables

The specific treatment of categorical features in machine learning models creates a potential issue for their interpretation via the Shapley interpreter. Here we propose a better way of dealing with them build upon the axioms presented in the previous chapter and show its desirable results on a simple example.

Categorical features are defined by having a finite number of discrete values. An apparent example of such a feature might be education with values such as elementary, graduate, doctorate, etc. A number of commonly used machine learning models can only work with numerical features – regression and support vector machines, to name a few – categorical features are thus transformed to be processable by them. A widespread solution is transforming every categorical variable into a group of dummy features, each of those indicating the presence or absence of a particular categorical feature's value [16].

For instance, a categorical random variable c with $K \in \mathbb{N}$ possible values from the set $\{1, \ldots, K\}$ gets replaced by K dummy binary variables $D_c = \left\{c^{(k)}\right\}_{k\in\{1,\ldots,K\}}$, where

$$c^{(k)} \equiv \begin{cases} 1 & , c = k \\ 0 & , c \neq k \end{cases} \quad , k \in \{1, \ldots, K\}\,. \tag{2.1}$$

This method of transformation, however, brings about issues with the implementation of the Shapley interpreter as presented in the previous section. The interpreter cannot distinguish that the group of dummy features is in fact one feature and treats them all separately, calculating a value for every one of them.

Let's take another look at the coalitional function from Equation (2.4):

$$v^{\boldsymbol{x}}(S) \equiv \mathbf{E}_{\mathbf{P}_{S^c}}\left[\, \mathfrak{f}_{\boldsymbol{x}_S}(X_{S^c}) \,\right] - \mathbf{E}_{\mathbf{P}_N}\left[\, \mathfrak{f}(X_N) \,\right] \tag{2.2}$$

Under this framework a strict subset of D_c might be a part of coalition S – this subset is then considered as known – while the rest is not and its values are randomly estimated. This approach entirely omits the information, that the features in D_c constitute one single feature, and should thus enter coalitions all at once.

In Section 1.6 we have defined the amalgamated T-game, where players from a group $T \subseteq N$ are merged together to form a single new player denoted by $\mathring{T}$. The coalitional function is then redefined to assure that all players in T enter coalitions simultaneously:

$$v^{T}(S) \equiv \begin{cases} v(S) & , \mathring{T} \notin S, \\ v\left(S \setminus \left\{\mathring{T}\right\} \cup T\right) & , \mathring{T} \in S, \end{cases} \tag{2.3}$$

where $S \subseteq N \setminus T \cup \left\{\mathring{T}\right\}$. To employ this idea and solve the problem of categorical variables described above, we define groups of dummy features for every categorical feature and then amalgamate each of these groups into one player describing the corresponding categorical feature as a whole through its Shapley value $\varphi^{\text{Sh}}_{\mathring{T}}$.

We have seen through the study of game theory axioms that summing up Shapley values of individual dummy features – as is often the recommended practice – will not produce this result, since we cannot generally claim that

$$\sum_{i \in T} \varphi^{\text{Sh}}_i(v) = \varphi^{\text{Sh}}_{\mathring{T}}\left(v^{(T)}\right). \tag{2.4}$$

The Shapley value is generally only efficient when $T = N$, which is a pointless case of only one categorical feature without the need for interpretation. We thus cannot use the old approach to obtain the value $\varphi^{\text{Sh}}_{\mathring{T}}$. Moreover, it would not be advantageous, as the amalgamated T-game can greatly reduce the number of features for the interpreter and speed up the calculation. The Banzhaf value is efficient when $|T| = 2$, which is, again, an uninteresting case of a binary variable that does not need to be dummy encoded.

Example 29. An often used approach to interpreting categorical features is summing up the individual Shapley values of their dummy features. We have argued above why this is not ideal, here we present a simple example showing that this method produces wrong results and compare it with the newly proposed method.

Table 2.1: Dataset, model, and their dummy encoded counterparts

x_1	x_2	$\mathfrak{f}(x_1, x_2)$
1	A	1
1	B	1
1	C	1
0	A	1
0	B	0
0	C	0

$\longrightarrow$

x_1	$x_2^{(A)}$	$x_2^{(B)}$	$x_2^{(C)}$	$\mathfrak{f}'\left(x_1, x_2^{(A)}, x_2^{(B)}, x_2^{(C)}\right)$
1	1	0	0	1
1	0	1	0	1
1	0	0	1	1
0	1	0	0	1
0	0	1	0	0
0	0	0	1	0

Table 2.1 shows a dataset with 2 features, binary x_1 and categorical x_2 with values $\{A, B, C\}$, and the outcome of a classification function

$$\mathfrak{f}\left(x_1, x_2\right) \equiv \left(x_1 == 1\right) \vee \left(x_2 == \text{A}\right). \tag{2.5}$$

The second table shows the same dataset, but with dummy encoded features $\left\{x_2^{(\text{A})}, x_2^{(\text{B})}, x_2^{(C)}\right\}$. For simplicity, we denote Shapley values of features in the first dataset as φ_{x_1} and φ_{x_2}, and as $\varphi'_{x_1}, \varphi'_{x_2^{(\text{A})}}, \varphi'_{x_2^{(\text{B})}}$, and $\varphi'_{x_2^{(\text{C})}}$ for the second dataset.

Let us focus on the first datapoint where $x_1 = 1, x_2 = \text{A}$ and find Shapley values for the two features. It can easily be shown that the two players are symmetric and their values are equal to $\left(\varphi_{x_1}, \varphi_{x_2}\right) = \left(1/6, 1/6\right)$.

We'd expect symmetry to hold between the features in dummy encoded dataset as well, since this is just a natural extension and should not affect the way they are interpreted in any way. However, Shapley values for this set are equal to $\left(\varphi'_{x_1}, \varphi'_{x_2^{(\text{A})}}, \varphi'_{x_2^{(\text{B})}}, \varphi'_{x_2^{(\text{C})}}\right) = (1/8, 1/8, 1/24, 1/24)$, and if we calculate the value for φ'_{x_2} the usual way by summing up the values of dummy players, we get

$$\varphi'_{x_2} = \sum_{k \in \{\text{A,B,C}\}} \varphi'_{x_2^{(k)}} = \frac{5}{24}, \tag{2.6}$$

which does not equal φ'_{x_1}. We would hence be mislead and conclude that features x_1 and x_2 have a different impact on the output of the function.

On the other hand, using the T-game and grouping dummy features $x_2^{(\text{A})}, x_2^{(\text{B})}, x_2^{(\text{C})}$ into $\mathring{T}$, we get $\left(\varphi'_{x_1}, \varphi'_{\mathring{T}}\right) = (1/6, 1/6)$ – the correct answer.

2.4 Sampling Shapley values

We have already discussed the problems of exponential complexity, here we propose an algorithm to deal with this issue. We will return back to the previous chapter and use Algorithm 1.1 for sampling of the Shapley interpreter, adjusting it to our needs.

First, let us transform the Shapley value from Equation (2.6) into its random-order form:

$$\varphi_i^{\text{Sh}}\left(v^{x}\right) = \frac{1}{n!} \sum_{\pi \in \Pi(N)} \Delta_i^{x}\left(\text{Pre}^i\left(\pi\right)\right), \tag{2.1}$$

where $\Pi\left(N\right)$ is the set of all permutations of N and $\text{Pre}^i\left(\pi\right)$ denotes the set of features preceding i in permutation π – the equivalence of this definition was discussed in Section 1.4.

The following algorithm differs from the formerly introduced Algorithm 1.1 in two main ways. Firstly, we have to take into account the concrete nature of the coalitional function in this example, which is itself an expected value,

$$v^{x}\left(S\right) \equiv \mathbf{E}_{\mathbf{P}_{S^c}}\left[\, \mathfrak{f}_{x_S}\left(X_{S^c}\right)\,\right] - \mathbf{E}_{\mathbf{P}_N}\left[\, \mathfrak{f}\left(X_N\right)\,\right] \tag{2.2}$$

with respect to the underlying distributions of features in $\mathcal{X}$. Computing this value precisely would soon prove to be computationally infeasible as well, forcing us to turn to yet another sampling procedure. We deal with this issue in subsec:sampling-kononenko.

Secondly, whereas in the above algorithm we took advantage of the fact that a single permutation π is capable of determining marginal contributions of every one of the features in N, here we have to

Algorithm 2.1 Sampling the Shapley interpreter for one feature

```
INPUT:
    -model $\mathfrak{f}$
    -feature $i$
    -data point $\boldsymbol{x}$
    -# of samples $m \in \mathbb{N}$
    -probability density functions of features in $\mathcal{X}$
OUTPUT:
    -estimate $\hat{\varphi}_i$ of $i$'s value in the Shapley interpreter
$\hat{\varphi}_i \leftarrow 0$
for sample in $m$:
    randomly draw $\pi$ from $\Pi(N)$ with uniform probabilities
    randomly draw $\boldsymbol{y}$ from $\mathcal{X}$
    construct $z^+, z^-$:   $z_k^+ = z_k^- = \begin{cases} x_k & , k \in \mathrm{Pre}^i(\pi) \\ y_k & , k \in \left(\mathrm{Pre}^i(\pi)\right)^{\mathrm{C}} \end{cases}$
    specify value of feature $i$:   $z_i^+ = x_i$
    calculate marginal contribution of feature $i$:
        $\hat{\Delta}^{\pi,\boldsymbol{y}}\left(\mathrm{Pre}^i(\pi)\right) = \mathfrak{f}(z^+) - \mathfrak{f}(z^-)$
    $\hat{\varphi}_i \leftarrow \hat{\varphi}_i + \hat{\Delta}^{\pi,\boldsymbol{y}}\left(\mathrm{Pre}^i(\pi)\right)$
end
$\hat{\varphi}_i \leftarrow \frac{\hat{\varphi}_i}{m}$
```

consider that the calculation of marginal contributions involves evaluating $\mathfrak{f}$, the most computationally difficult part of the algorithm. It is therefore advantageous to choose cautiously which feature's value should be calculated in order to maximize accuracy of the estimates with a given number of samples. This is the topic of 2.4.2.

2.4.1 Sampling one feature

Algorithm 2.1 describes, in pseudocode, sampling of feature i's Shapley value. In each sample we draw a random vector $\boldsymbol{y}$ from the feature space together with a permutation π and set $\boldsymbol{y}$'s elements in indices from $\mathrm{Pre}^i(\pi)$ to be equal to corresponding elements of $\boldsymbol{x}$. We then estimate the marginal contribution of i to this permutation.

In another words, the sampling population in this case is the set of marginal contributions

$$P^{\boldsymbol{x}} = \left\{\hat{\Delta}^{\pi,\boldsymbol{y}}; \pi \in \Pi(N), \boldsymbol{y} \in \mathcal{X}\right\} \tag{2.3}$$

and we draw m samples $P_1, \dots, P_m$ from $P^{\boldsymbol{x}}$ at random with replacement according to the probability distributions of $\mathcal{X}$ and $\Pi(N)$[6]. The final estimate is then calculated as the sample mean, $1/m \sum_{k=1}^{m} P_k$.

Štrumbelj and Kononenko [55] analyze the properties of this approximation. They use datasets with a small number of features (enabling them to calculate their true Shapley values) and show that sufficient accuracy is achieved with less than 10000 samples per feature while using 1000 samples brings the absolute error down to around 0.05.

[6]In the case of Shapley value, this is, of course, a uniform distribution.

As we have seen in Section 1.7, the algorithm creates estimates that approximately follow the normal distribution with mean $\varphi_i^{\mathrm{Sh}}(v^{\boldsymbol{x}})$ and variance σ_i^2/m, where σ_i^2 is the population variance – these facts allow us to develop a second algorithm for distributing the total number of samples in a way that minimizes the expected error of the approximation.

2.4.2 Adaptive sampling

For a complete interpretation we need to repeat Algorithm 2.1 for every feature in N. Let us suppose that the maximum number of samples at our disposal is $M \in \mathbb{N}$: distributing this number naively and allocating M/n samples to every feature could cause redundant calculations, for instance, in the case of features not contributing to the model. We will now present a better criterion for distributing M, adaptive sampling. To simplify the notation in the following paragraphs, let us for now denote i's Shapley value from Equation (2.1) simply as φ_i and its approximation from Algorithm 2.1 as $\hat{\varphi}_i$, our goal is to minimize the sum of their squared differences. By m_i we denote the number of samples used for approximating the Shapley value of feature i.

In the preceding discussion we have seen that

$$\hat{\varphi}_i \approx \mathcal{N}\left(\varphi_i, \frac{\sigma_i^2}{m_i}\right)$$

and thus the following holds:

$$\hat{\varphi}_i - \varphi_i \approx \mathcal{N}\left(0, \frac{\sigma_i^2}{m_i}\right) \tag{2.4}$$

We know from probability theory that $\mathrm{E}\left[Z^2\right] = \mathrm{Var}\left[Z\right] + \mathrm{E}\left[Z\right]^2$ for any random variable Z and so, if we take $Z := (\hat{\varphi}_i - \varphi_i)$, use the linearity property of $\mathrm{E}\left[\cdot\right]$, and substitute values from Equation (2.4), we can write

$$\mathrm{E}\left[\sum_{i=1}^{n} (\hat{\varphi}_i - \varphi_i)^2\right] = \sum_{i=1}^{n} \frac{\sigma_i^2}{m_i}. \tag{2.5}$$

From now on, we distribute samples one by one and use Algorithm 2.1 with $m = 1$. We first take a portion of M and use it to calculate initial approximations for all the features[7], obtaining estimates of their variances $\hat{\sigma}_i^2$ as well. Then, in each step we wish to determine which feature should be selected to minimize the expected error of the approximation – the sum on the right hand side of Equation (2.5). To do so, let us suppose that up until the current iteration we have assigned $(m_1, \ldots, m_n)$ samples to the features $1, \ldots, n$. The sum $\sum_{i=1}^{n} \hat{\sigma}_i^2/m_i$ will be minimized by selecting feature $j \in N$ that satisfies

$$j = \operatorname*{arg\,max}_{k \in N} \left\{ \frac{\hat{\sigma}_k^2}{m_k} - \frac{\hat{\sigma}_k^2}{m_k + 1} \right\}, \tag{2.6}$$

or, in another words, whose contribution to the error will decrease the most before the next step. This is summarized in Algorithm 2.2.

This approximation decreases the exponential time complexity to $\mathcal{O}(n \cdot \mathcal{T}(\mathfrak{f}))$, where $\mathcal{T}(\mathfrak{f})$ is the time complexity of evaluating model $\mathfrak{f}$ at an arbitrary datapoint.

[7]In all of the examples in this thesis, this portion was arbitrarily set to $M/5$.

Algorithm 2.2 Adaptive sampling of all the features

```
INPUT:
    -model $f$
    -data point $x$
    -total # of samples $M \in \mathbb{N}$
    -# of samples $m_{\min} \leq M$ before adaptive sampling starts
OUTPUT:
    -estimates $\{\hat{\varphi}_i;\quad i \in N\}$ for prediction at $x$
$\hat{\varphi}_i \leftarrow 0$, $m_i \leftarrow 0$,     $\forall i \in N$
while $\sum_{i=1}^{N} m_i \leq M$:
    if $\sum_{i=1}^{N} m_i \leq m_{\min}$ :
        choose feature $j$ s.t.  $m_j \leq {m_{\min}}/{n}$
    else:
        choose feature $j$ that maximizes ${\hat{\sigma}_j^2}/{m_j} - {\hat{\sigma}_j^2}/{m_j+1}$
    $\hat{\Delta}_j$ =result of Algorithm 2.1 for feature $j$ with $m = 1$
    $\hat{\varphi}_j \leftarrow \hat{\varphi}_j + \hat{\Delta}_j$
    calculate new estimate of $\hat{\sigma}_j^2$
    $m_j \leftarrow m_j + 1$
end
$\hat{\varphi}_i \leftarrow {\hat{\varphi}_i}/{m_i}$,     $\forall i \in N$
```

Estimating variance The last remaining question is that of estimating the variances $\hat{\sigma}_i^2$ for each feature i. For the sake of text clarity, in the following paragraphs we consider m samples $(x_1, \ldots, x_m)$ and we wish to compute their population variance σ_m^2. Let $\bar{x}_m$ denote the sample mean: $\bar{x}_m = {1}/{m} \sum_{i=1}^{m} x_i$.

As Chan, Golub, and Leveque [11] and Knuth [29] note, using the naive calculation

$$\sigma_m^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \bar{x}_m)^2 ,$$

or equivalently

$$\sigma_m^2 = \frac{1}{m} \left(\sum_{i=1}^{m} x_i^2 - \frac{1}{m} \left(\sum_{i=1}^{m} x_i \right)^2 \right),$$

could lead to computational rounding errors due to subtracting two large numbers, especially when m is large and variance is small. As it happens, this is the case for the Shapley interpreter, where we use many samples and the algorithm converges quite quickly.

Welford [62] suggested a different iterative approach, updating the quantity

$$S_m = \sum_{i=1}^{m} (x_i - \bar{x}_m)^2 = m \cdot \sigma_m^2 \tag{2.7}$$

through recurrence relations

$$\bar{x}_m = \bar{x}_{m-1} + \frac{1}{m} (x_m - \bar{x}_{m-1})$$

and

$$S_m = S_{m-1} + (x_m - \bar{x}_{m-1})(x_m - \bar{x}_m)$$

with initial values set naturally as

$$\bar{x}_1 = x_{1,}$$
$$S_1 = 0.$$

This method is less likely to fail due to numerical errors and, since we estimate our approximations iteratively one by one, it is also convenient for our use.

2.5 Follow-up work

Several authors have taken up the Shapley interpreter and used it as a basis for further interpretation methods. Here we provide their short survey.

Notably, Lundberg and Lee[33] defined a class of additive feature attribution methods (similarly to Equation (2.7)) and showed that both LIME and the Shapley interpreter fall under this class. This allowed them to state that there is only one unique solution within this class of methods – the Shapley value – satisfying three desirable properties. They then provide a concrete definition of LIME (see Equation (2.1)) whose outcome is identical to that of the Shapley interpreter, but estimates values for all features simultaneously resulting in a faster approximation.

Frye, Feige, and Rowat [22] propose relaxing the symmetry axiom and using quasivalues for interpreting a model instead. They call this framework ASV – Asymmetric Shapley Values. The argument against symmetry lies in the fact that it can conceal important influences on the model and relaxing this axiom helps with incorporating knowledge of any causal relations among the features.

For instance, we can order features into two groups so that those with a causal precedence before the latter come in the first group and the rest in the second (one such example might be age which largely determines marital status). Through this we can discover the true effect the features in the first group have on the model without incorporating their implicit impact on the latter features. We can also gain interesting insights into the model by ordering the features the other way around: allowing only permutations where, e.g., sex comes before test scores while interpreting a model for college admission, answers the question of how much does candidates' sex matter after their test results are known.

Implementing ASV is simple, in Section 1.7 we have presented Algorithm 1.1 for sampling of general quasivalues. The only difference from the implementation of the Shapley interpreter is then specifying a concrete probability distribution over $\Pi(N)$ that encompasses any causal knowledge we might have about the data.

Another contribution of the article lies in suggesting a data generating process that creates data "on manifold". The authors note that the Shapley interpreter (see Algorithm 2.1) places no constraints on the perturbed data points, causing them to be unrealistic and forcing the model to predict on vectors that have no true relation to the training dataset.

This fact is further developed by Slack et al. [52], who have realized that data points used for interpreting a model are fundamentally different from those used in real scenarios and trained a neural network discerning between the two. This effectively allowed them to create a model that behaves fairly when it judges it is being evaluated in an interpretation environment, whilst hiding an unfair model used otherwise.

Aas, Jullum, and Løland [1] have tried to counter this issue and came with a novel process for data generation that takes dependencies among features into account. They provide several possibilities for improvement of sampling from the marginal distribution in Equation (2.6), ranging from fitting a Gaussian distribution to the data to Monte Carlo methods. This however further increases the computational complexity of the Shapley interpreter, as conditional probability distributions have to be calculated, or at least approximated, at every step of the sampling procedure.

Chapter 3

Correlation analysis of value operators

In the previous chapter we defined the Shapley interpreter, an interpretation technique built upon coalitional game theory and its Shapley value. In Chapter 1, we saw that the Shapley value is not the only possible solution and we examined properties of other value operators. Here we compare the results of the standard Shapley value with the Banzhaf value on several benchmark datasets.

Banzhaf value, defined in Definition 23 by

$$\varphi_i^{\mathrm{B}}(v) \equiv \sum_{S \subseteq N \setminus \{i\}} \frac{1}{2^{n-1}} \Delta_i^v(S), \tag{3.1}$$

are a special case of semivalues where each coalition is drawn with the same probability. It satisfies the dummy player axiom and it is additive and symmetric, however, contrary to the Shapley value, it is not efficient. This explains its common use as index of power describing the size of each player's impact on the game.

The use of the Banzhaf value in the context of machine learning explainability has thus far been only sparsely explored. In an article by Somol, Grim, and Pudil [53], the authors propose a method for feature selection by averaging marginal contributions across a random subsample of coalitions: essentially employing the Banzhaf value, albeit the connection to game theory is not explicitly emphasized in the article. Recently in an article by Patel, Strobel, and Zick [42], Banzhaf value and its interaction indices are proposed as the optimal solution minimizing approximation error between the model and k-degree polynomials of the form

$$g^k(\boldsymbol{x}) = I_0 + \sum_{S \subset N; |S| \leq k} I(S) \prod_{j \in S} x_j. \tag{3.2}$$

The Shapley and Banzhaf value operators differ in the way they ascribe probabilities to coalitions $S \subseteq N \setminus \{i\}$. Whereas the Banzhaf value uses a uniform distribution over $2^{N \setminus \{i\}}$, the Shapley value uses a probability distribution defined by

$$\mathrm{P}(S) = \frac{1}{n \cdot \binom{n-1}{|S|}}. \tag{3.3}$$

We see that this is equal to first choosing a cardinality of S from $\{0, \ldots, n-1\}$ with uniform probabilities $1/n$, and then picking a random coalition with that cardinality out of the $\binom{n-1}{|S|}$ possibilities. Coalitions with small or large (close to n) cardinalities are thus picked more often and they play a large role when calculating the final value. This can be closer seen in Figure 3.1, where the probabilities of picking a concrete coalition of a given cardinality are compared.

Freixas [20] theoretically analyzed the ordinal equivalence of semivalues (i.e. whether the resulting players' values have the same order – more about this later) and defined subclasses of games where they

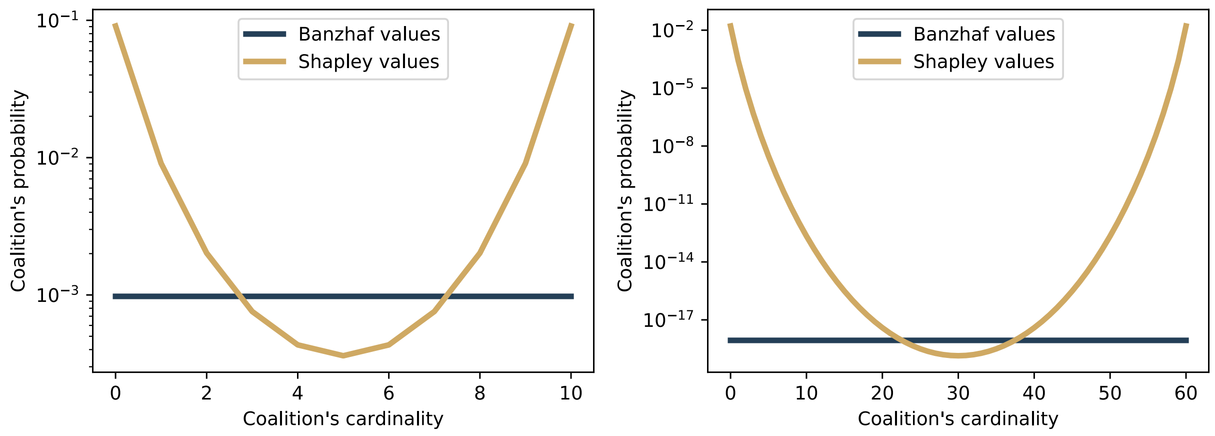


Figure 3.1: Comparison of $\mathrm{P}(S)$ for S of a given cardinality for two games with $n = 11, 61$. Notice that the probabilities differ in many orders of magnitude.

are equivalent. Nevertheless, we cannot generally use his results to an arbitrary interpretation game defined in Section 2.2 and we have therefore carried out an extensive comparison described in the following sections.

We have trained and tested several models on several datasets, these are all presented and described in Section 3.1. In Section 3.2 we introduce the chosen method of comparison, the final results and their discussion are in Section 3.3.

In Section 2.4 we presented an algorithm for sampling the results of the Shapley interpreter, its extension to the Banzhaf value is straightforward with the use of Algorithm 1.2. We have implemented all of the algorithms using the standard libraries of Python version 3.7.6.

3.1 Datasets and models

We have prepared 3 datasets and trained 3 models on each of them. Here we first present and describe the datasets, then we turn to the models, their hyperparameters, and their performance.

3.1.1 Datasets

We have used 3 datasets from the classic UCI Machine Learning Repository [17] with an increasing number of features:

Titanic Dataset contains information about 891 passengers of the Titanic with the goal of predicting their survival. The dataset includes 8 features, among those are, for instance, age, sex, ticket class, and number of relatives on board. We also include the title of each passenger as a feature. The dataset contains three categorical features, we have treated these via dummy encoding and amalgamation as described in Section 2.3.

Breast Cancer Wisconsin (Diagnostic) Dataset describes characteristics of cell nuclei present in images of fine needle aspirates of breast mass. It contains 569 instances with 30 features, all numeric, for instance radius, symmetry, etc. The task is to classify the instances as either malignant or benign.

Optical Recognition of Handwritten Digits Dataset contains 1797 images of handwritten digits from 0 to 9 with the task of correctly classifying these digits. Each image is a matrix of 8x8 where each element is an integer between 0 and 16. All of the 10 classes roughly contain the same number of instances.

3.1.2 Models

Each of the aforementioned datasets has been randomly split into a training set and a test set, the latter one containing 20% of the overall number of instances and we have trained 3 nonlinear models on the training sets. These models and their settings are described in the following paragraphs while the obtained performance on the test sets is summarized in Table 3.1. we have implemented the models with the help of Python's open source SCIKIT-LEARN library version 0.22.1 [43] and XGBOOST library version 1.0.2 [12]. For hyperparameters not explicitly mentioned, we use their default setting.

Feedforward neural network (NN) A multilayered perceptron with 4 hidden layers containing 50, 50, 50, and 30 neurons, respectively. We use adaptive learning rate and ReLU activation functions in each of the hidden layers. The network is optimized with the adam solver.

Gradient boosted decision trees (GBDT) An ensemble of 100 gradient boosted decision trees, each at most 5 layers deep. The learning rate is set to 0.1.

Support Vector Machine (SVM) A support vector machine with a Gaussian kernel, regularization parameter set to 1, and gamma equal to $1/\text{\# features}$.

3.2 Framework for comparison

With the prepared datasets and trained models, we are ready to define the corresponding coalitional games, compute their Shapley and Banzhaf values, and analyze the values' similarities/differences. For each of the dataset/model pairs we randomly pick 200 instances, hence defining 200 coalitional games, and perform the following analysis.

We use the algorithms described above to obtain the Shapley and Banzhaf values for each of these games, here denoted with vectors $\boldsymbol{\varphi}^{\mathrm{S}}$ and $\boldsymbol{\varphi}^{\mathrm{B}}$. The i-th elements of these vectors then describe feature i's impact on the model.

Since the Banzhaf values are not normalized, we opt for rank correlations and compare the order of features under the two different solutions, rather than comparing the magnitudes of the vectors' elements. We first calculate two rank vectors, $\boldsymbol{r}^{\mathrm{S}}$ and $\boldsymbol{r}^{\mathrm{B}}$, where r_i^{S} is equal to the rank of φ_i^{S} in $\boldsymbol{\varphi}^{\mathrm{S}}$ and identically for $\boldsymbol{r}^{\mathrm{B}}$. In case of ties, we assign distinct ranks to each of the elements based on their order in the vector. We then use two nonparametric measures of rank correlation describing the similarity between $\boldsymbol{r}^{\mathrm{S}}$ and $\boldsymbol{r}^{\mathrm{B}}$.

Table 3.1: Models' accuracy on training and test sets.

	NN		GBDT		SVM	
	train	*test*	*train*	*test*	*train*	*test*
titanic	85.7%	87.7%	91.3%	87.2%	82.7%	86.0%
breasts	92.7%	96.5%	100.0%	95.6%	91.4%	93.9%
digits	100.0%	98.3%	100.0%	96.1%	99.6%	98.9%

Spearman's ρ is calculated via the standard formula of Pearson's correlation coefficient, where we substitute the rank variables $\boldsymbol{r}^{\mathrm{S}}$ and $\boldsymbol{r}^{\mathrm{B}}$ for $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively:

$$\rho \equiv \frac{\sum_i \left((x_i - \overline{\boldsymbol{x}}) (y_i - \overline{\boldsymbol{y}}) \right)}{\left(\sum_i (x_i - \overline{\boldsymbol{x}})^2 \sum_i (y_i - \overline{\boldsymbol{y}})^2 \right)^{1/2}}. \tag{3.1}$$

Kendall's τ is "based on the principle that if there is association between the ranks in $\boldsymbol{r}^{\mathrm{S}}$ and $\boldsymbol{r}^{\mathrm{B}}$, then if we arrange ranks $\boldsymbol{r}^{\mathrm{S}}$ in ascending order (so that $r_i^{\mathrm{S}} = i$) the r_i^{B} should show an increasing trend if there is positive association and a decreasing trend if there is negative association". (Cited from Sprent and Smeeton [54] with modified symbols for variables) For each $i \in \{1, \ldots, n-1\}$ and $j > i$, we count the number of positive and negative differences $r_j^{\mathrm{B}} - r_i^{\mathrm{B}}$ and denote them with n_+ and n_- respectively. The Kendall correlation coefficient is then calculated by

$$\tau \equiv \frac{n_+ - n_-}{\frac{1}{2} n (n-1)}. \tag{3.2}$$

Since the denominator is equal to the total number of possible pairs, hence equal to $n_+ + n_-$, Kendall's τ gives values from $[-1, 1]$ and their interpretation is identical to that of the standard correlation coefficients.

3.3 Results

We have calculated Spearman's ρ and Kendall's τ for each of the 200 instances of every model/dataset pair from Table 3.1. The results are summarized in Tables 3.2, 3.3, and 3.4 for datasets Titanic, breast, and digits, respectively. For each dataset and model, the tables show distributions of the 200 correlation coefficients between the Shapley and Banzhaf values.

Quite expectedly, the correlations are lower when we interpret datasets with a larger number of features. The Titanic dataset, containing 8 features, achieves high correlations falling mostly between 0.9 and 1.0. The breast and digits datasets, containing 30 and 64 features, respectively, achieve similar, somewhat smaller, correlations with means around 0.8.

Overall the results are not supportive of the claim, that the Shapley and Banzhaf values yield significantly different interpretations. Correlations between them are consistently high across all the tested datasets and models.

This, we believe, justifies further research into the use of Banzhaf values (or any other value operator) in the context of model interpretation. It has been suggested by Lundberg and Lee [33] that the Shapley values are the optimal solution due to their efficiency property, however, lately this claim has been challenged by Patel, Strobel, and Zick [42], where the 2-efficiency property described in Section 1.6 is upheld as desirable for interpretation. We have shown, that the two values are closely linked together, albeit with slightly differing results. This small difference is something that is yet to be fully understood by the underlying theory and axiomatization.

Furthermore, whereas the Shapley value is usually used for its efficiency property allowing for a fair distribution of the game's value, the Banzhaf values is used as a measurement of power. The above similarities essentially permit the use of the Shapley value as a method for feature selection, and similarly, allow for the Banzhaf value to be viewed as an approximation of the Shapley interpreter.

These are, of course, nonconclusive results, as different datasets or different models could show bigger differences between the two values. We also acknowledge that artificial environments with synthetic datasets and models could be created, where the outcomes of the above comparison would be significantly different and suggest a wide contradiction between the Shapley and Banzhaf values. However, this was not the goal of this chapter as we have preferred to test the hypothesis on datasets that resemble those standardly used in real-life applications.

Table 3.2: Percentiles of correlation coefficients for the Titanic dataset.

	NN		GBDT		SVM	
	ρ	τ	ρ	τ	ρ	τ
mean	0.96	0.91	0.98	0.95	0.96	0.91
min	0.61	0.50	0.76	0.57	0.67	0.50
5%	0.86	0.78	0.90	0.79	0.78	0.64
10%	0.90	0.79	0.95	0.86	0.83	0.71
25%	0.95	0.86	0.98	0.93	0.95	0.86
50%	0.98	0.93	1.00	1.00	0.98	0.93
75%	1.00	1.00	1.00	1.00	1.00	1.00
90%	1.00	1.00	1.00	1.00	1.00	1.00
95%	1.00	1.00	1.00	1.00	1.00	1.00
max	1.00	1.00	1.00	1.00	1.00	1.00

Table 3.3: Percentiles of correlation coefficients for the breasts dataset.

	NN		GBDT		SVM	
	ρ	τ	ρ	τ	ρ	τ
mean	0.82	0.77	0.89	0.81	0.94	0.94
min	0.46	0.43	0.35	0.39	0.70	0.71
5%	0.60	0.57	0.65	0.58	0.80	0.81
10%	0.68	0.63	0.70	0.65	0.84	0.86
25%	0.76	0.71	0.88	0.78	0.88	0.89
50%	0.84	0.78	0.93	0.84	0.99	0.97
75%	0.90	0.84	0.96	0.88	1.00	1.00
90%	0.94	0.89	0.97	0.91	1.00	1.00
95%	0.97	0.91	0.98	0.92	1.00	1.00
max	0.99	0.97	0.99	0.94	1.00	1.00

Table 3.4: Percentiles of correlation coefficients for the digits dataset.

	NN		GBDT		SVM	
	ρ	τ	ρ	τ	ρ	τ
mean	0.83	0.71	0.84	0.74	0.80	0.69
min	0.67	0.54	0.67	0.56	0.57	0.44
5%	0.74	0.60	0.75	0.63	0.68	0.57
10%	0.75	0.61	0.76	0.65	0.70	0.59
25%	0.79	0.65	0.80	0.69	0.75	0.64
50%	0.85	0.72	0.85	0.75	0.81	0.69
75%	0.89	0.77	0.89	0.78	0.85	0.75
90%	0.91	0.79	0.92	0.82	0.90	0.79
95%	0.92	0.82	0.93	0.84	0.91	0.81
max	0.95	0.85	0.97	0.91	0.94	0.86

Chapter 4

Brain activity analysis

In this final chapter we present an extensive illustration of using the Shapley interpreter for analyzing a real-world medical dataset. We have obtained measurements of brain activity from a group of patients and use this data for discovering connections among major brain regions.

We start out by a closer description of the data, then use Section 4.2 to talk about our model and how it is trained. We analyze this model through the Shapley interpreter in Section 4.3 and summarize the results in Section 4.4. Due to the similarities between the Shapley and Banzhaf values described in the previous chapter, we have decided not to include the Banzhaf value in the analysis.

4.1 Data

The dataset contains measurements of brain activity by functional magnetic resonance imaging (fMRI). In short, fMRI measures activity by detecting changes in blood flow in various regions of the brain, a method based on the fact that "when an area of the brain is in use, blood flow to that region also increases." [63]

In total 84 subjects were studied, each being measured for 240 time steps. The dimensionality of the sample was reduced by averaging the signal within 90 known anatomical regions of the brain. We have concatenated the data from individual subjects in order to obtain longer time series, resulting in a dataset with 90 features and 20160 instances. We have further normalized the data so that each feature has mean equal to zero across all the instances and standard deviation equal to one. The time series for brain region number 1 can be seen at Figure 4.1.

We set out to use this data to discover and describe connections among the 90 regions, or, in other words, how strongly do the regions influence each other as the time series develops. In order to do this we will train a model predicting the evolution of the series, described in more detail in the following section.

4.2 Model

The goal of the data analysis is finding a model that best predicts the next step in the time series. To avoid a common pitfall of time series modelling, where the model simply predicts the next step to be exactly the same as the current one (which is, indeed, a passable solution constituting a local minimum out of which it might prove difficult to escape), we have turned the problem from regression into binary classification. Instead of predicting the exact value of the time series, we changed the task to predicting whether the series will rise or drop in value between the current and the next step (hence predicting the

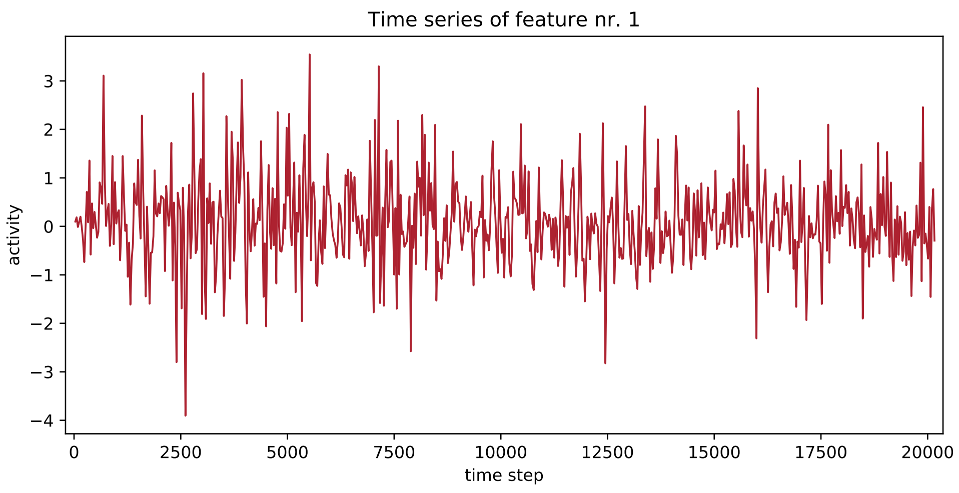


Figure 4.1: Example time series for region nr. 1. Only every 30th instance is shown for better readability.

sign of the first order derivative w.r.t. time). We believe that this crude approximation is still sufficient for our task of finding connections among the regions.

Let us set up some basic notation. We denote our dataset by $\mathcal{D} = (\boldsymbol{x}^t, \boldsymbol{y}^t)_{t=1}^{m}$, where $m = 20160$ is the total number of instances, and $\boldsymbol{x}^t \in \mathbb{R}^n$ is a vector describing the $n = 90$ regions in each time step t. The target vector $\boldsymbol{y}^t \in \mathbb{R}^n$ is defined as

$$y_i^t \equiv \begin{cases} 1 & , x_i^{t+1} \geq x_i^t \\ 0 & , x_i^{t+1} < x_i^t \end{cases} \tag{4.1}$$

for each region $i \in \{1, \ldots, n\}$. The last vector $\boldsymbol{y}^m$, where the true target value is unknown, has been arbitrarily set to be a vector of ones. The target variable is evenly distributed with 50.45% of samples being positive when averaged across all regions and instances, with similar numbers for each region individually.

We have used the first 17660 instances as a training set for estimating the model's parameters and the following 1500 instances as a validation set that has been held out from the training process and used to approximate the model's performance on previously unseen data. To prevent accidentally tailoring the model to the validation set, we have hidden the last 1000 instances altogether and only used them for the final verification once the best model has been found. We call these 1000 instances the test set.

As a metric of performance we have used estimates of binary cross entropy. For two probability distributions p and q, cross entropy is defined by

$$H(p, q) \equiv -\operatorname{E}_p [\log(q)] ,$$

which, if we assume discrete probabilities, can be rewritten as

$$-\sum_x p(x) \log q(x) .$$

Let us denote by $p\left(y_i^t\right)$ a model's prediction for any datapoint from $\mathcal{D}$. Since we are dealing with a binary problem, cross entropy can be rewritten in a simpler form, hence defining our loss function:

$$\mathcal{L} \equiv -\frac{1}{N} \sum_{t=1}^{m} \sum_{i=1}^{n} \left[y_i^t \cdot \log\left(p\left(y_i^t\right)\right) + \left(1 - y_i^t\right) \cdot \log\left(1 - p\left(y_i^t\right)\right) \right] . \tag{4.2}$$

This represents the loss value across the entire dataset $\mathcal{D}$, scope of the sums is adjusted for its subsets. We wish to find a model that minimizes loss on the validation set.

4.2.1 Model architecture

Due to the time series nature of our data, we have found it suitable to use recurrent neural networks (RNNs). These differ from the simpler feedforward neural networks (FFNNs) by also including feedback connections among the layers, making them better at processing sequences. Our sources for this section were reviews of RNNs by Lipton, Berkowitz, and Elkan [32] and Sutskever [59], and a detailed description of LSTM networks by Gers, Schmidhuber, and Cummins [23].

Whereas FFNNs do not incorporate the concept of time and assume all the datapoints to be independent of each other, RNNs contain a hidden state that is combined with new observations through an intricate nonlinear function. This can, to put it simply and perhaps somewhat misleadingly, be likened to the concept of memory, making RNNs especially useful in contexts where the temporal dimension plays an important role.

Long short-term memory (LSTM) networks are a particular type of RNNs first introduced by Hochreiter and Schmidhuber [27] that are easier to train. Explaining the mathematics behind these models is beyond the scope of this text, we refer the interested reader to references in the margin.

After an initial period of trial-error testing, we have settled on the following architecture with promising results to be explored further in finer detail. The network consists of 4 layers:

1. An LSTM layer with ReLU activation function and sigmoid activation function for the recurrent step.

2. A second LSTM layer with configuration identical to the first one.

3. A densely connected feedforward layer with ReLU activation function.

4. An output layer with sigmoid activation function returning probabilities of the target variable being equal to one for each output feature.

Due to the two back-to-back LSTM layers, the first one is set to return whole sequences instead of only their last member.

There is a number of hyperparameters that need to be specified before the training can start, here we highlight them and use the next subsection to talk about them in concrete numbers. Firstly the width of each of the first three layers needs to be set. We shall refer to the number of units in the two LSTM layers by `lstm1_units` and `lstm2_units` respectively, and to the number of units in the dense layer by `dense_units`.

We use a dropout hyperparameter to constrain overfitting on the training data. The LSTM layer allows for two kinds of dropout: one for the layer's input units – which is similar to the standard dropout layer – and one for the connections among the layer's recurrent units. We use both of these dropouts on both of the LSTM layers, however, to simplify the model we use a single number for all of them. We will refer to this number as `dropout`.

For training the network and minimizing the loss function, we use the Adaptive Moment Estimation (Adam) algorithm [28] for stochastic gradient descent. Adam iteratively adapts its learning rate based on estimates of first and second order moments of the loss function, we have fixed its β_1 and β_2 decay parameters to be equal to 0.9 and 0.999 respectively. The initial learning rate is another hyperparameter of our model, we will refer to it by `lr_init`.

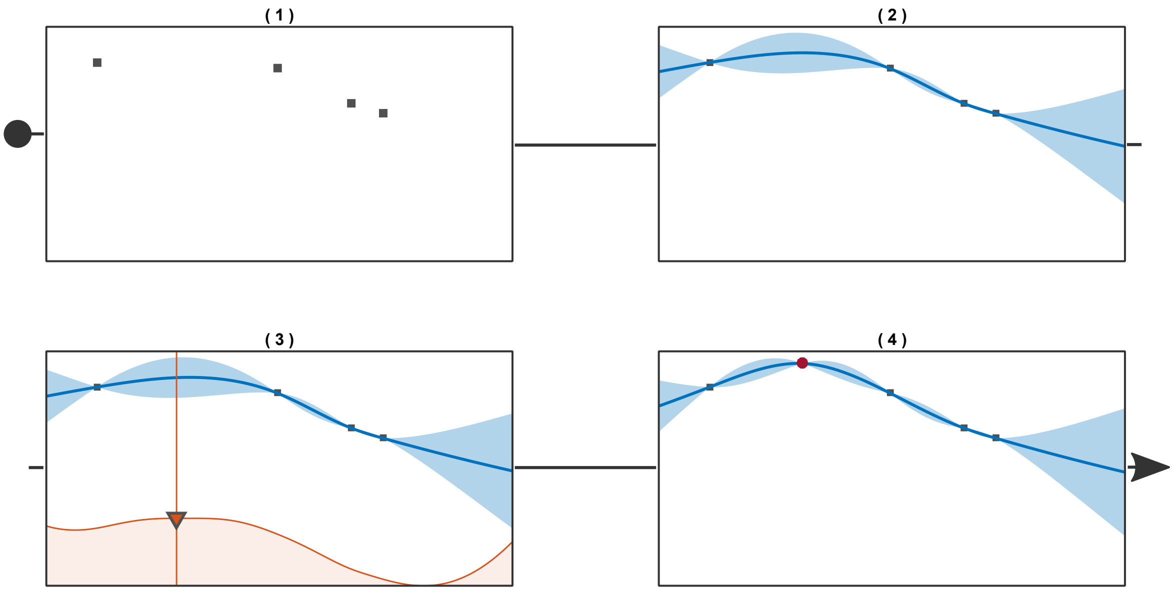


Figure 4.2: Bayesian optimization schema for maximization

`batch_size` refers to the number of datapoints we use simultaneously for training of the network at each step. We stop the training process whenever validation loss hasn't decreased for 50 consecutive epochs and restore the weights from the epoch with the best performance.

Last but not least, we wanted to test the dependence of the model's performance on its input shape. We set a discrete hyperparameter `time_steps`, controlling the number of steps back in time that are included in the input data. In other words, we wanted to see how far back in time do we need to look to be able to achieve a good prediction for the next time step.

We have implemented all of the above with the help of the Keras deep learning API [13] version 2.2.4 and the TensorFlow library [2] version 1.14.0. Any of the parameters not explicitly mentioned have been set to their default values.

4.2.2 Hyperparameter tuning

This gives us in total 7 hyperparameters to tune. To find their optimal setting we have employed the powerful framework of Bayesian optimization via Gaussian processes.

4.2.2.1 Theory

Bayesian optimization is useful for finding the optima of a function whenever the function itself is expensive to enumerate and we wish to minimize the number of times we do so. In our context, the function to optimize is the performance of the network, where we set the hyperparameters defined in the previous section to specific values and observe the resulting loss from Equation (4.2) on the validation set, which serves as our metric for minimization. In each iteration of the optimization process, the function is estimated based on its previous evaluations, and a new point (hyperparameter setting) of inquiry is specified. We have demonstrated this process schematically for a function with one parameter in Figure 4.2. For more information about Gaussian processes we recommend a comprehensive monography by Rasmussen and Williams [44], Bayesian optimization is neatly summarized in an article by Brochu, Cora, and Freitas [8].

Let us denote the function to optimize as $f(\boldsymbol{x}) : \mathbb{R}^p \to \mathbb{R}$, where $p \in \mathbb{N}$ denotes the dimensionality of our input space (7 in our case). To estimate the function, we interchange it with a Gaussian process with mean and covariance functions (undetermined for now) $\mu(\boldsymbol{x})$ and $C(\boldsymbol{x}, \boldsymbol{x}')$ respectively. This gives us a distribution over the set of functions: for a set of $M \in \mathbb{N}$ datapoints $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ we can draw a random function from the Gaussian process via

$$(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_M)) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{C}), \tag{4.3}$$

where $\mathcal{N}$ stands for the normal distribution, $\boldsymbol{\mu} = (\mu(\boldsymbol{x}_1), \ldots, \mu(\boldsymbol{x}_M))$, and $\boldsymbol{C} \in \mathbb{R}^{M \times M}$ is a matrix defined by $C_{ij} = C\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$.

The mean function of the process is standardly set to be equal to zero for all inputs to reflect our lack of knowledge without any measurements. There are many possibilities for the covariance function, we have used the exponential function

$$C(\boldsymbol{x}, \boldsymbol{x}') \equiv \sigma^2 \exp\left[-\frac{1}{2s^2}\left\|\boldsymbol{x} - \boldsymbol{x}'\right\|^2\right]. \tag{4.4}$$

Here the parameters σ and s control the magnitude and scale of the estimate and $\|\cdot\|$ is the L^2 norm.

Equation (4.3) tells us how to create prior estimates. To incorporate any already existing measurements of $f(\boldsymbol{x})$, we have to calculate several multivariate integrals. Luckily, in the case of regression, these integrals are analytically tractable and thus easy to evaluate. Here we only present the final result and refer the reader to the literature in margins for a greater detail.

Let us again use a set of M datapoints $X = \{\boldsymbol{x}_i\}_{i=1}^M$, now together with their known function values $\boldsymbol{f} = \{f(\boldsymbol{x}_i)\}_{i=1}^M$. To create posterior estimates of function value for a new datapoint $\boldsymbol{x}^*$, we calculate matrix $\boldsymbol{C}$ in the same way as before, vector $\boldsymbol{C}^* \in \mathbb{R}^M$ of covariances among $\boldsymbol{x}^*$ and datapoints from X, and use the following result:

$$f(\boldsymbol{x}^*) \mid \boldsymbol{f}, X, \boldsymbol{x}^* \sim \mathcal{N}\left(\boldsymbol{C}^{*T}\boldsymbol{C}^{-1}\boldsymbol{f}, C(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{C}^{*T}\boldsymbol{C}^{-1}\boldsymbol{C}^*\right) \tag{4.5}$$

This gives us a probability distribution of function values for every input vector (hyperparameter setting). Moreover, this distribution is by definition Gaussian, hence for every $\boldsymbol{x} \in \mathbb{R}^p$ we obtain estimates of the expected value and variance for $f(\boldsymbol{x})$, further denoted as $m(\boldsymbol{x})$ and $s(\boldsymbol{x})$ respectively. We use these estimates to find the most promising point for the next measurement.

We do this by defining an ACQUISITION FUNCTION that quantifies the *potential of improvement* for every $\boldsymbol{x}$ and evaluate $f(\boldsymbol{x})$ in the point of biggest potential – this is represented by the orange line in Figure 4.2.

Once again, there are many possibilities – we chose to measure the expected improvement:

$$\mathrm{EI}(\boldsymbol{x}) \equiv \mathrm{E}\left[\max\{\tau - f(\boldsymbol{x}), 0\}\right], \tag{4.6}$$

where $\tau = \min_{\boldsymbol{x} \in X} f(\boldsymbol{x})$ denotes the current minimum and the expected value is calculated w.r.t. the distribution of $f(\boldsymbol{x})$. This can be rewritten as

$$\mathrm{EI}(\boldsymbol{x}) = \begin{cases} (\tau - m(\boldsymbol{x}))\,\Phi\left(\frac{\tau - m(\boldsymbol{x})}{s(\boldsymbol{x})}\right) + s(\boldsymbol{x})\,\phi\left(\frac{\tau - m(\boldsymbol{x})}{s(\boldsymbol{x})}\right) & s(\boldsymbol{x}) \neq 0 \\ 0 & s(\boldsymbol{x}) = 0, \end{cases} \tag{4.7}$$

where $\Phi(\cdot)$ stands for cumulative distribution function of the standard Gaussian distribution and $\phi(\cdot)$ is its density. This form of the acquisition function compactly shows, that it takes into account both the expected mean value and the uncertainty of each estimate – this is referred to as the exploration-exploitation compromise.

In each step of the optimization we thus have to inverse the matrix $\boldsymbol{C}$. By doing so, we however create a powerful Bayesian estimate through which we can find the function's optimum faster than by using the much easier random search.

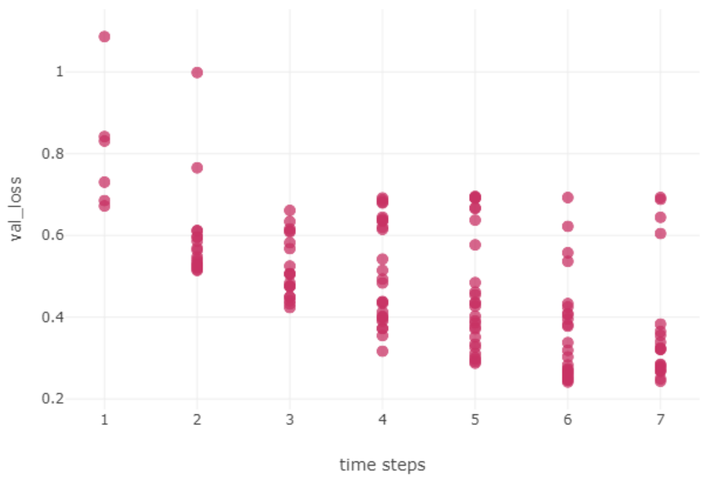


Figure 4.3: Dependence of validation loss on `time_steps`

4.2.2.2 Praxis

We used Bayesian optimization to find the optimal hyperparameter setting when predicting future values for region nr. 1. Table 4.1 summarizes constraints of the hyperparameter space with minimal and maximal possible values for each hyperparameter and shows optimal values of the best model, which has achieved loss of 0.240 and 89.9% accuracy on the validation set, loss 0.275 and 88.4% accuracy on the test set, and loss 0.273 and 88.2% accuracy on the training set.

We started by evaluating the function for 15 randomly selected hyperparameter settings and then ran the optimization for 60 iterations, evaluating the function twice in each of them before producing a new estimate. We have implemented the algorithms for optimization with help of the GPyOpt library [3] version 1.2.6.

Given the nature of our data, we find these results to be sufficient. We have achieved similar performance on all of the three datasets, thus avoiding overfitting and finding genuine relations among the features.

The most interesting hyperparameter is `time_steps`, controlling the number of steps back in time available to the model. Figure 4.3 shows a scatterplot containing all the trained models and showing dependence of validation loss on the hyperparameter. The loss decreases constantly until time step $t-6$, where it seems to have reached a plateau. To constrain the total number of features, we chose not to increase the number of time steps any further.

Table 4.1: Hyperparameter ranges and their optimal values

	`time_steps`	`lstm1_units`	`lstm2_units`	`dense_units`	`dropout`	`lr_init`	`batch_size`
min	1	150	150	200	0.4	$^1/_{10000}$	1500
max	7	500	500	400	0.9	$^4/_{1000}$	3000
best	6	152	414	368	0.549	$^1/_{1000}$	1854

4.2.3 Model performance

Rather than training a single model encompassing predictions for all the 90 regions, we have decided to train 90 unique models, one for predicting the next step in each region individually. We believe this will help with creating diverse models that are tuned specifically to their corresponding regions and effective at discovering connections among them.

We have not repeated the Bayesian optimization to find optimal hyperparameters for each of the 90 models individually – that would prove to be too computationally expensive – but we have used the optimum for region nr. 1 from Table 4.1. This rests on the assumption that all of the regions are similar and that the same architecture can be used to model all of them. This assumption is, we believe, justified by low variance of achieved loss and accuracies of the models.

The performance is summarized in Table 4.2. We have constantly achieved satisfactory results without overfitting the models on either the training or the validation set.

4.3 Using the Shapley interpreter

With the models prepared, we turn to the Shapley interpreter to analyze them and search for the connections among regions. There are, however, a couple more things we need to specify before we can run the final analysis.

In each step of the interpretation, we generate random datapoints for out-of-coalition features based on their marginal distribution (remember Equation (2.4)). Here we have assume independence among the features, mainly for simplicity and a substantial decrease in computational complexity. To further decrease variance in our sampling, we approximate each feature's distribution by its mean (which is zero, due to the normalization described above). That is, out-of-coalition features are set to be equal to zero, while the rest keeps their values from the original datapoint.

We have sampled datapoints from $\mathcal{D}$ using a uniform distribution and ran the interpreter for each of the 90 models. In total, we have obtained local Shapley values for *750* datapoints, where for each of them we have run the interpreter for 54000 iterations, using adaptive sampling. Each of the Shapley values tells us the impact its corresponding feature has on prediction for the next time step, for one particular datapoint. However, we are interested in a global estimation of the impact – hence we define the following aggregation, further referred to as GLOBAL SHAPLEY VALUES.

Let us for any given datapoint $\boldsymbol{x} \in \mathcal{D}$ denote its local Shapley values by a vector $(\varphi_i(\boldsymbol{x}))_{i=1}^{540}$, where 540 is the total number of features (90 regions times 6 steps back in time). We then calculate the global aggregation as an average of absolute values of the local Shapley values,

$$\varphi_i^{\mathrm{G}} \equiv \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} |\varphi_i(\boldsymbol{x})| \tag{4.1}$$

Table 4.2: Performance of the 90 models

	train set		validation set		test set	
	loss	*accuracy*	*loss*	*accuracy*	*loss*	*accuracy*
best	0.227	90.5%	0.205	91.5%	0.215	92.0%
mean	0.275	87.9%	0.250	89.0%	0.257	88.9%
worst	0.310	86.0%	0.294	86.9%	0.322	85.6%

4.4 Results & discussion

With the framework for analysis complete, we can finally move on to see its results and conclusions.

4.4.1 Connections of regions nr. 1, 57, and 50

First we want to look for regions that influence the state of brain activity for region nr. 1 (picked without any particular reasons). Figure 4.4a shows a graphical representation of its global Shapley values.

We will encounter this type of figures repeatedly, it is thus worth taking a bit of time to understand this one thoroughly. The rows discern between the $n = 90$ regions of the brain, while each column belongs to a particular step in time. For instance, the cell in the 6th row and 2**nd** column describes the influence the brain activity in time step $t - 2$ of region nr. 6 has on the prediction for region nr. 1. The color of the cell then describes the magnitude of this influence.

Now back to Figure 4.4a. We can readily obtain a couple of interesting observations:

- Quite expectably, it is region nr. 1 itself that has the biggest predicting power.
- The influence grows with increasing time lag, with time steps $t - 4$ and $t - 5$ achieving the largest values. Time step $t - 6$ does have a big influence, but stops the increasing trend. This is in accord with Figure 4.3.
- Next to region nr. 1, there are several other regions with a strong connection. Here we list regions with average influence (mean global Shapley values when averaged across all time steps) larger than 0.02, in decreasing order: 1, 57, 2, 19, 11, and 68.

In their seminal article, Štrumbelj and Kononenko[57] suggest using the standard deviation of samples for aggregation of local Shapley values. We have tested this approach as well and the results are almost identical to those above, hence we have decided to stick with our definition of the aggregation due to its easier interpretability.

The connection between regions nr. 1 and 57 seems promising, we wanted to explore this finding further and verify it by reversing the direction: next we look at important predictors for region nr. 57 in Figure 4.4b.

Similar inferences can be drawn from this side of the figure. Once again, the region itself is the biggest predictor with time steps $t-4$ and $t-5$ being the most influential. The list of regions with largest values is 57, 58, 1, 2, and 69.

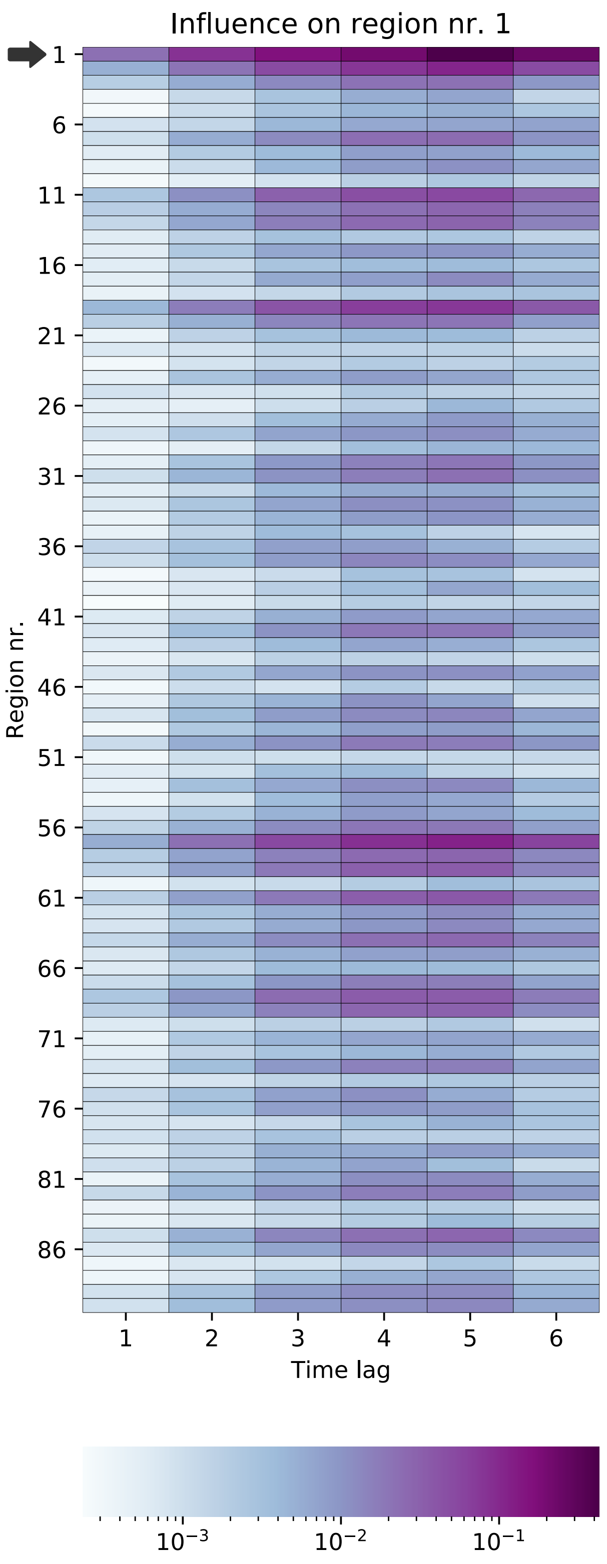
The significant connection between regions nr. 1 and 57 has thus been confirmed: on average, region nr. 57 changes the prediction for region nr. 1 by 0.06 (in absolute value) and, the other way around, region nr. 1 changes the prediction for region nr. 57 by 0.05. To put this into context, the average value of region nr. 1 for prediction there is 0.20, while for region nr. 57 this value is 0.17.

We carried out analogous analysis for region nr. 50, again picked arbitrarily. The results are shown in Figure 4.5. Here we see a group of strong predictors surrounding the region, namely these are, in descending order of importance, regions nr. 50, 49, 52, 46, 48, 51, and 45.

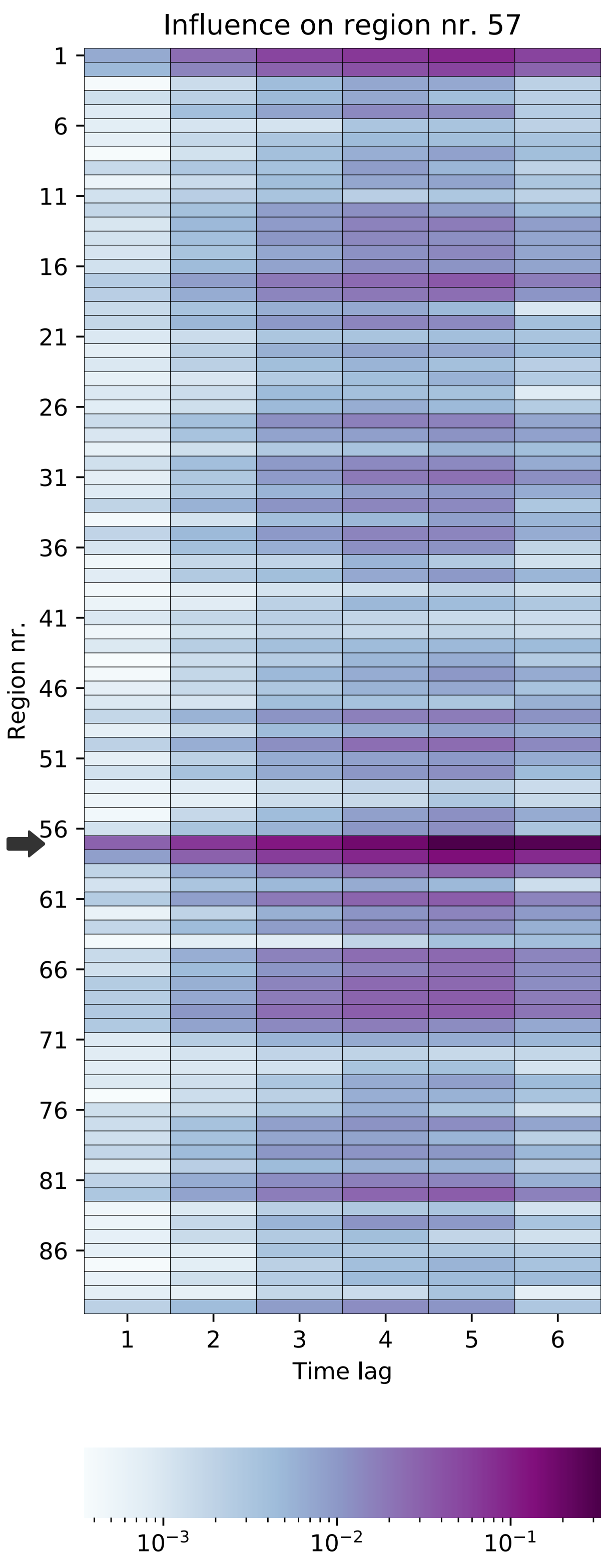
This has led us to the idea of discovering all such interconnected groups of regions.

4.4.2 Connected groups

So as not to overwhelm the reader, we show the rest of the results in a shorter format, creating an encompassing clustered graph containing all of the regions.



(a) Region nr. 1



(b) Region nr. 57

Figure 4.4: Means of absolute Shapley values

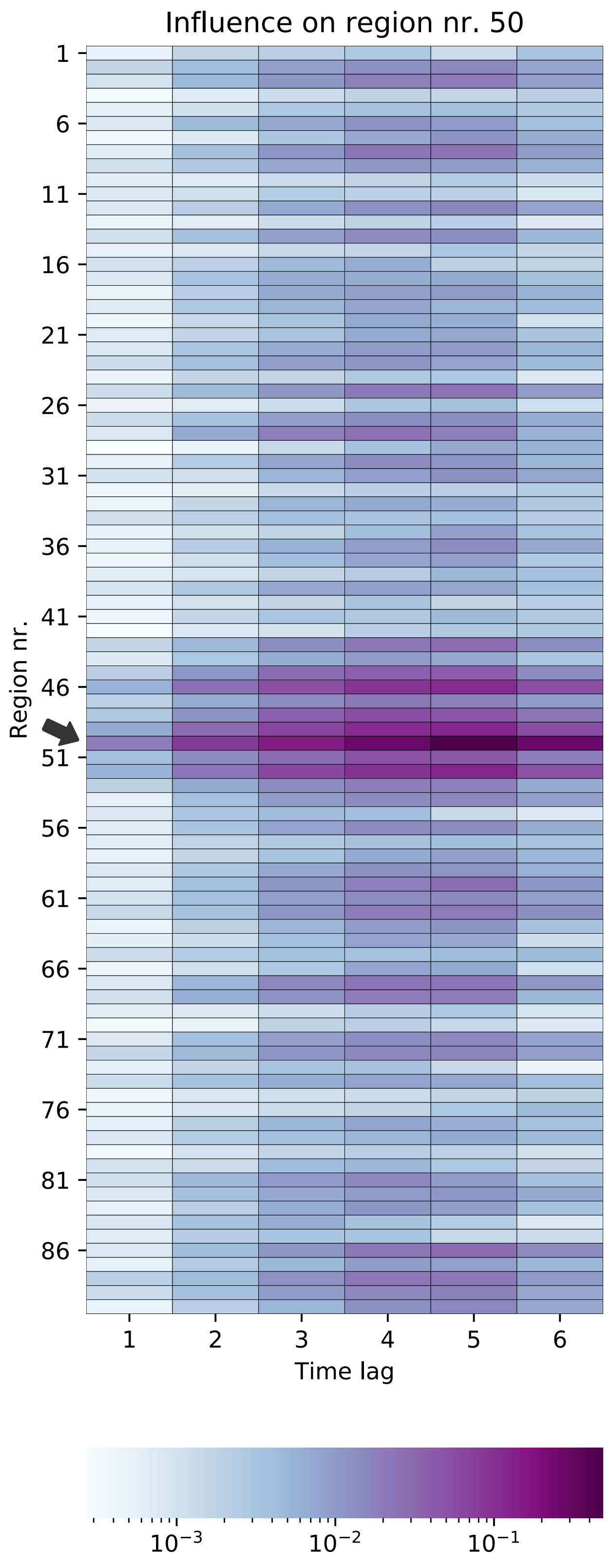


Figure 4.5: Means of absolute Shapley values for region nr. 50

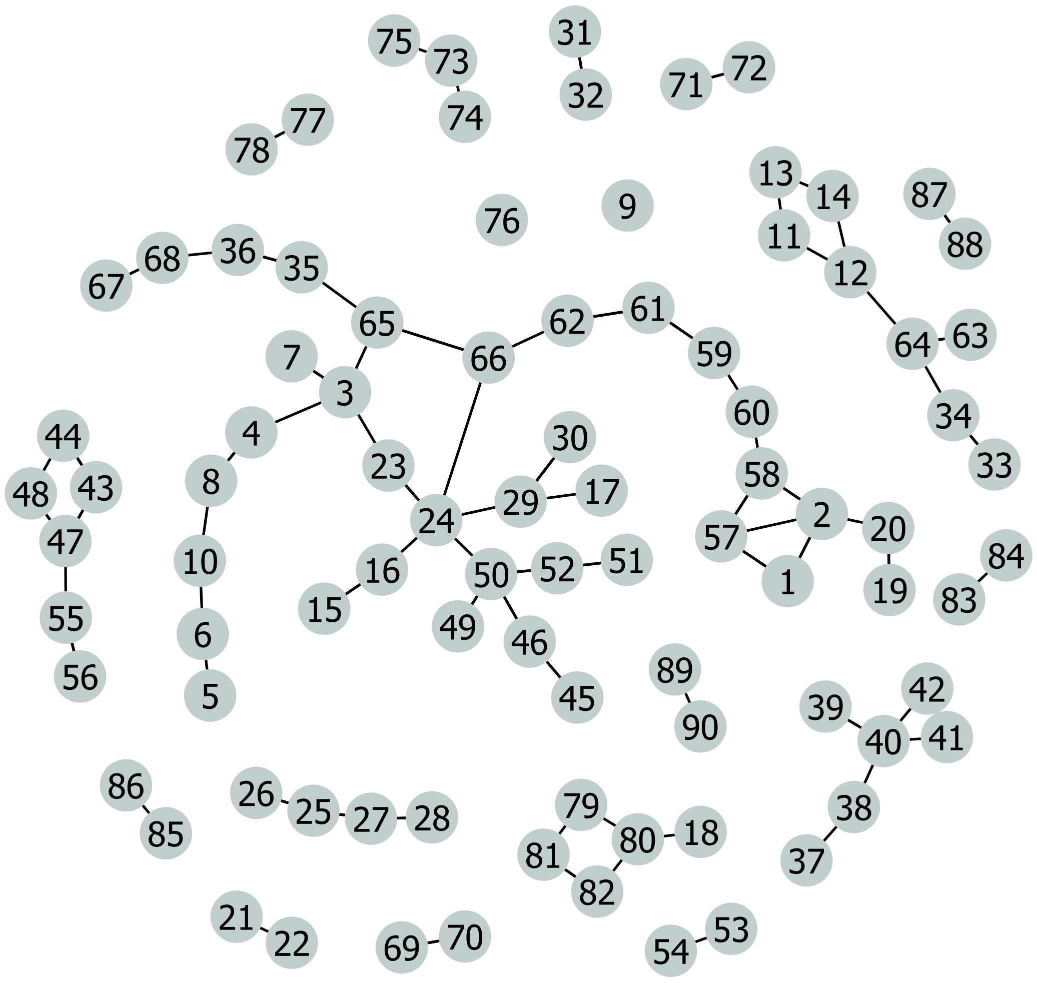


Figure 4.6: Clustered groups of interconnected regions.

We have used the 90 models from Subsection 4.2.3 and repeated the above analysis to discover groups of regions that influence each other. For every region we have noted every other region whose average influence is at least 25% as big as the influence of the original region onto itself.

The results can be seen in Figure 4.6. We have decided not to use directed edges to make the figure easier to read, we include an edge whenever at least one of its ends has a large influence on the other one. Loops are also omitted as they would have to be at every vertex and hence make the graph overfull. Note that the distance among vertices plays no particular role.

We see that the regions are quite neatly clustered into a couple of disjoint groups. Often these groups contain regions with similar numbers (73-74-75), there are however many connections between regions with numbers far from each other (3-65, 24-66, or the aforementioned 1-57). A large number of the regions show only one, or none at all, significant influence. Here the influence of each of the regions onto itself is too strong and it pushes all the other regions below the specified threshold of 25%.

The use of the Shapley value comes with an important advantage of model-agnosticity. We could use an arbitrary model, run the analysis described above, and obtain analogous results. The same framework could be used for any other data, however complex or nonlinear.

Conclusion

This thesis revolves around coalitional game theory and primarily its use for interpreting statistical models that are otherwise difficult to comprehend.

By rigorously defining the interpretation method and studying the axioms of coalitional game theory, we have been able to suggest a novel approach to interpreting models with categorical features. Moreover, we have shown on a simple counterexample that the current method of doing so yields erroneous results.

After implementing the presented sampling algorithms, we have set out to compare the now standard Shapley value with a different solution concept from coalitional game theory, the Banzhaf value. Neither in theory, nor in our comprehensive applied comparison on real world datasets, were we able to find any significant disparities. These results, although preliminary, suggest that the two values can be used interchangeably in the context of interpretation.

In the last chapter we have thoroughly analyzed a dataset describing the brain activity of a group of human subjects. We have used Bayesian optimization to train an intentionally complex model for predicting the evolution of the time series and then used the interpretation techniques to gain insights into connections among brain regions.

In conclusion, we believe that the Shapley value forms a powerful framework for interpretation and see potential in future research. We hope that this text is one of many that will attempt to connect the interpreter with the foundations of coalitional game theory, out of which new ideas and methods could emerge. There are interesting unanswered questions concerning specific definitions of the coalitional game, comparison of results for classification and regression (i.e. simple vs. normal games), or incorporation of the Shapley value for interactions among features.

Bibliography

[1] Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values, 2019.

[2] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[3] The GPyOpt authors. Gpyopt: A bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`, 2016.

[4] John F. Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.*, 19:317, 1964.

[5] Jason Barr and Francesco Passarelli. Who has the power in the eu? *Mathematical Social Sciences*, 57(3):339 – 366, 2009.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[7] Olga N. Bondareva. Some applications of linear programming methods to the theory of cooperative games. In *Problemy Kybernetiki*, volume 10, pages 119–139, 1963.

[8] Eric Brochu, Vlad Cora, and Nando Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, 12 2010.

[9] André Casajus. Amalgamating players, symmetry, and the banzhaf value. *International Journal of Game Theory*, 41(3):497–515, 2012.

[10] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726 – 1730, 2009.

[11] Tony F. Chan, Gene H. Golub, and Randall J. Leveque. Algorithms for computing the sample variance: Analysis and recommendations. *The American Statistician*, 37(3):242–247, 1983.

[12] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[13] François Chollet et al. Keras. `https://keras.io`, 2015.

[14] Filip K. Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0210–0215, 2018.

[15] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv: Machine Learning*, 2017.

[16] Norman R. Draper and Harry Smith. *"Dummy" Variables*, chapter 14, pages 299–325. John Wiley & Sons, Ltd, 2014.

[17] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[18] Pradeep Dubey. On the uniqueness of the shapley value. *International Journal of Game Theory*, 4(3):131–139, 1975.

[19] Ernest Fokoué. Model selection for optimal prediction in statistical machine learning. *Notices of the American Mathematical Society*, 67:1, 02 2020.

[20] Josep Freixas. On ordinal equivalence of the shapley and banzhaf values for cooperative games. *International Journal of Game Theory*, 39:513–527, 10 2010.

[21] Hannah Fry. *Hello World: Being Human in the Age of Algorithms*. W. W. Norton Company, September 2019.

[22] Christopher Frye, Ilya Feige, and Colin Rowat. Asymmetric shapley values: incorporating causal knowledge into model-agnostic explainability, 2019.

[23] Felix Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12:2451–71, 10 2000.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[25] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5), August 2018.

[26] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. Lemna: Explaining deep learning based security applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 364–379. Association for Computing Machinery, 2018.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[28] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[29] Donald E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., USA, 1997.

[30] Martin Kopp, Tomáš Pevný, and Martin Holeňa. Anomaly explanation with random forests. *Expert Systems with Applications*, 149, 2020.

[31] Ehud Lehrer. An axiomatization of the banzhaf value. *International Journal of Game Theory*, 17(2):89–99, 1988.

[32] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning, 2015.

[33] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[34] Michael Maschler, Eilon Solan, and Shmuel Zamir. *Game Theory*. Cambridge University Press, 2013.

[35] Dov Monderer and Dov Samet. Chapter 54 variations on the shapley value. In R. J. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications*, volume 3, pages 2055 – 2076. Elsevier, 2002.

[36] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

[37] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[38] Cathy O'Neil. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown Publishing Group, 2016.

[39] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press. MIT Press, 1994.

[40] Guillermo Owen. *Game Theory*. Emerald Group Publishing Limited, 2013.

[41] Guillermo Owen and Lloyd S. Shapley. Optimal location of candidates in ideological space. *International Journal of Game Theory*, 18(3):339–356, 1989.

[42] Neel Patel, Martin Strobel, and Yair Zick. High dimensional model explanations: an axiomatic approach, 2020.

[43] Fabian Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[44] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005.

[45] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “why should i trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’16, page 1135–1144. Association for Computing Machinery, 2016.

[46] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat Mach Intell*, 1(1):206–215, 2019.

[47] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[48] Lloyd S. Shapley. A value for n-person games. In H. Kuhn and A. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, 1953.

[49] Lloyd S. Shapley. On balanced sets and cores. *Naval Research Logistics Quarterly*, 14(4):453–460, 1967.

[50] Lloyd S. Shapley and Martin Shubik. A method for evaluating the distribution of power in a committee system. *American Political Science Review*, 48(3):787–792, 1954.

[51] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR, 06–11 Aug 2017.

[52] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods, 2019.

[53] Petr Somol, Jiří Grim, and Pavel Pudil. Fast dependency-aware feature selection in very-high-dimensional pattern recognition. In *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pages 502–509, 10 2011.

[54] Peter Sprent and Nigel Smeeton. *Applied Nonparametric Statistical Methods*. 09 2000.

[55] Erik Štrumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11:1–18, 01 2010.

[56] Erik Štrumbelj and Igor Kononenko. A general method for visualizing and explaining black-box regression models. In *Adaptive and Natural Computing Algorithms*, pages 21–30, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[57] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665, Dec 2014.

[58] Erik Štrumbelj and Igor Kononenko. *Explaining the Predictions of an Arbitrary Prediction Model: Feature Contributions and Quasi-nomograms*, pages 139–157. Springer International Publishing, Cham, 2018.

[59] Ilya Sutskever. *Training recurrent neural networks*. University of Toronto Toronto, Ontario, Canada, 2013.

[60] Diego Varela and Javier Prado-Domínguez. Negotiating the lisbon treaty: Redistribution, efficiency and power indices. *AUCO Czech Economic Review*, 6:107–124, 07 2012.

[61] Robert J. Weber. *Probabilistic values for games*, pages 101–119. Cambridge University Press, 1988.

[62] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[63] Wikipedia contributors. Functional magnetic resonance imaging — Wikipedia, the free encyclopedia, 2020. [Online; accessed 14-June-2020].

[64] Eyal Winter. Chapter 53 the shapley value. In R. J. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications*, volume 3, pages 2025 – 2054. Elsevier, 2002.