Quantitative Research Methods

Explainable AI with Shapley Values

Advanced module A2 — optional self-study

Prof. Dr. Christoph Weisser

HSBI

Summer Semester 2026

Where this module fits

This is one of four **optional advanced modules** that extend the core course. Nothing here is exam material for the main chapters — but everything here is built from tools you already own.

You already know	From	Used here for
Flexibility vs. interpretability	Ch. 2	why accurate models need explaining
Boosting, variable importance	Ch. 8	the black box we explain; the benchmark
Deep networks	Ch. 10	the archetypal black box (same machinery)
The 2^p subset count	Ch. 0b, Ch. 6	the cost of exact Shapley computation
Seeded simulation, resampling	Ch. 5	background samples, Monte-Carlo error

Today's question

Chapter 8 gave us a boosted model that predicts well; Chapter 10 gave us networks nobody can read. This module answers: **for one prediction, how much did each feature contribute — exactly, with a guarantee?**

Contents

Optional and advanced material is collected in the **appendix** at the end of the deck.

Notation in this module

Symbol	Meaning
p	number of features (game-theory: players); here $p = 6$
$\mathcal{P} = \{1, \ldots, p\}$	the grand coalition: all players together
$S \subseteq \mathcal{P} \setminus \{j\}$	a coalition (subset of players) not containing j; $\lvert S \rvert$ its size
$v(S)$	value function: the worth achieved by coalition S alone
φ_j	Shapley value of player / feature j
$f(x)$	the fitted model's prediction at input x
x_S	the explained instance's values on the features in S
$X_{\bar{S}}$	the random features *outside* S
φ_0	baseline: $E[f(X)]$, the average prediction
$b^{(1)}, \ldots, b^{(B)}$	background sample of B rows (here $B = 100$)
π, $\mathrm{pre}_\pi(j)$	a random ordering of the players; those placed before j in π
m	number of sampled orderings in Monte-Carlo estimation

Sources and references

Primary textbook (course background)

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An Introduction to Statistical Learning, with Applications in Python.* Springer. — Chapters 8 and 10 supply the models we explain.

Module sources

- Shapley, L. S. (1953). A value for *n*-person games. *Contributions to the Theory of Games II*, Princeton UP.
- Štrumbelj, E., & Kononenko, I. (2014). Explaining prediction models . . . with feature contributions. *Knowledge and Information Systems*.
- Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *NeurIPS* (the SHAP paper).
- Lundberg, S. M., et al. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence* (TreeSHAP).
- Aas, K., Jullum, M., & Løland, A. (2021). Explaining individual predictions when features are dependent. *Artificial Intelligence*.
- Molnar, C. (2022). *Interpretable Machine Learning*, 2nd ed. (free online book).

Learning objectives

By the end of this module you will be able to:

- **Distinguish** local from global explanations, and say what a feature attribution should promise.
- **Define** the Shapley value of a cooperative game and **verify** its four axioms on concrete numbers.
- **Translate** a fitted model into a game via the value function $v(S)$, and **defend** a choice between marginal and conditional expectations.
- **Implement** exact Shapley values from scratch (2^p enumeration) and their Monte-Carlo approximation, and **check** the efficiency axiom numerically.
- **Read** waterfall, global-importance and dependence displays, and **compare** them with Chapter 8's permutation importance.
- **Avoid** the three classic pitfalls: off-manifold evaluation, causal over-reading, and attribution instability.

Roadmap of this module

From a parlour game to a regulator-ready explanation

1. Why accurate models need explaining; local vs. global.
2. Cooperative games and the Shapley value: formula, axioms, a fully worked three-player game.
3. From games to models: the value function $v(S)$ and the marginal-vs.-conditional choice.
4. Computation: exact 2^p enumeration by hand, Monte-Carlo sampling, KernelSHAP and TreeSHAP.
5. Reading the output: waterfall, mean $|\varphi_j|$, dependence plots — against permutation importance.
6. Pitfalls: correlated features, causation, instability — and what industry regulation demands.

Where this module is used in industry

Sector	Concrete application	Which decision it drives
Consumer credit	Adverse-action reasons for each declined loan application	Which "principal reasons" appear on the legally required notice
Insurance	Justifying a boosted pricing model factor by factor	Whether a rating factor survives actuarial and regulatory review
Healthcare	Which vitals and labs drove a patient's risk score	Whether the clinician trusts, overrides or escalates the alert
Fraud & AML	Explaining why a transaction was flagged	Investigator triage; evidence in the case file
Tech / marketing	Debugging ranking and churn models: what drove a bad score	Fix the data pipeline vs. retrain the model
Model risk / audit	Global driver analysis and drift monitoring	Model approval, re-validation, decommissioning

In industry — The common thread

In every row the model is a *tree ensemble or neural net* that beats the interpretable alternative — and a human downstream who must *account* for a single prediction. Shapley values are the de-facto standard answer because they come with a guarantee: the attributions **sum exactly** to the prediction.

Industry case in depth: explaining a credit decline

In industry — The setting: a boosted model decides — the law wants reasons

A bank scores consumer-loan applications with a gradient-boosted model — hundreds of trees over ~ 40 features. US regulation (ECOA / Regulation B) requires the bank to give each declined applicant the **principal reasons**; the EU AI Act classifies credit scoring as high-risk with transparency duties. "The model said so" is not a legal answer.

How Shapley values enter the pipeline

For each declined applicant x the bank computes attributions $\varphi_1, \ldots, \varphi_p$ that split $f(x) - E[f(X)]$ across features. The most negative φ_j's are mapped to standardised reason codes ("debt-to-income ratio too high"). The *background* is chosen deliberately — e.g. approved applicants — so the reasons answer "why you, compared to those who were approved".

Two honest caveats — both taught in this module

The reasons describe the *model*, not the applicant's causal reality — they are not financial advice. And with correlated features (income, debt, limits) the split between related reasons is partly a modelling choice, so banks fix background, seed and model version in their model-risk documentation.

Outline

Accurate models nobody can read

Chapters 8 and 10 ended with a trade: **accuracy up, readability gone**.

Our running model (built in this module)

A `GradientBoostingRegressor` predicting **log salary** on the `Hitters` data ($n = 263$ players after dropping missing salaries, $p = 6$ features) reaches **training** $R^2 = 0.965$ — using 100 trees of depth 3. That is roughly 800 leaf rules added together: no coefficient table to read, no single split to point at.

What we can no longer answer (yet)

- A player asks: *why does the model pay me less than him?*
- A manager asks: *which statistics drive salary overall?*
- An auditor asks: *does the model use the feature you claim it ignores?*

Takeaway

We refuse to trade the accuracy back. Instead we add an **explanation layer** on top of the

Local vs. global — and what an attribution must promise

Local explanation

For **one** instance x: how much did each feature value push *this* prediction $f(x)$ away from the average prediction? *(the declined applicant, the flagged transaction)*

Global explanation

For the **whole** model: which features matter most *on average* across all instances? *(model validation, driver analysis)*

What a feature attribution should mean

Numbers $\varphi_1, \ldots, \varphi_p$, one per feature, such that: (i) they **sum exactly** to $f(x) - E[f(X)]$ — no leftover, no double count; (ii) a feature the model **ignores** gets 0; (iii) two features used **identically** get the same number. These wishes will become the Shapley *axioms* — and they pin down a *unique* answer.

Takeaway

Shapley values are **local** objects; averaging $|\varphi_j|$ over many instances then buys the global

Exercise A2.1 — Local or global? [Concept]

Exercise

For each question, say whether it needs a **local** or a **global** explanation, and what the attribution numbers should sum to (if anything):

1. "Why was *my* loan declined?"
2. "Which features should we monitor for drift in production?"
3. "The model predicts salary \$743k for this veteran while the average prediction is \$349k — what accounts for the gap?"
4. "Can we drop `PutOuts` from the pipeline without losing accuracy?"

Solution — Exercise A2.1

Solution

Method. Ask *whose* prediction is being discussed: one instance ⇒ local; the model as a whole ⇒ global.

1. **Local.** One applicant, one prediction. The attributions should sum to $f(x) - E[f(X)]$: the gap between *this* score and the average score.
2. **Global.** A property of the model over its input distribution; mean $|\varphi_j|$ across instances is a natural monitor. No sum constraint — it is an average of magnitudes.
3. **Local**, and the sum is pinned down exactly: on the log scale the φ_j must add up to $\log 743 - \log 349 \approx 0.756$. (This is our running example — we will compute those six numbers exactly.)
4. **Global** — but note the trap: a small mean $|\varphi_j|$ says the model *currently* leans little on `PutOuts`, not that the feature is useless; after refitting without it, other correlated features may absorb its role.

Common mistake

Answering a local question with a global tool: telling the declined applicant "income is the most important feature of our model" explains the *model*, not *their* decision.

Outline

Cooperative games: players, coalitions, value

A **cooperative game** is a pair $(\mathcal{P}, v)$:

Definition

- $\mathcal{P} = \{1, \ldots, p\}$: a set of p **players**;
- $v : 2^{\mathcal{P}} \to \mathbb{R}$ with $v(\varnothing) = 0$: a **value function** giving the worth $v(S)$ that any **coalition** $S \subseteq \mathcal{P}$ can achieve on its own.

The question: how should the grand coalition's worth $v(\mathcal{P})$ be **divided fairly** among the players?

How to read this

"Fair" must reflect *contribution*: a player who adds a lot to every coalition they join deserves more. The natural raw material is the **marginal contribution** $v(S \cup \{j\}) - v(S)$ — what j adds when joining the coalition S. The difficulty: this number *depends on* S.

In industry — Cost allocation: the original use

A three-player game with concrete numbers

Three specialists found a consultancy: **A** (modeller), **B** (engineer), **C** (designer). Quarterly revenue (in 1 000 €) for every possible team:

The full value function

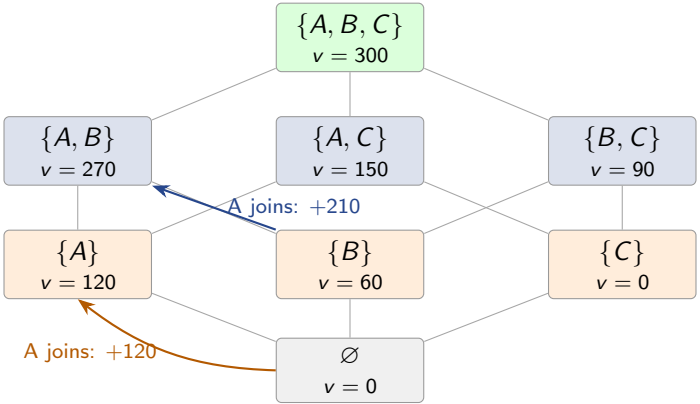
S	$\varnothing$	$\{A\}$	$\{B\}$	$\{C\}$	$\{A,B\}$	$\{A,C\}$	$\{B,C\}$	$\{A,B,C\}$
$v(S)$	0	120	60	0	270	150	90	300

C earns *nothing* alone, yet every team gains 30 when C joins it ($150-120$, $90-60$, $300-270$): C's design work only pays off on top of a product. A and B together show *synergy*: $270 > 120 + 60$.

Takeaway

Naive splits fail: equal thirds (100 each) ignores that A alone is worth 120; paying stand-alone values $(120, 60, 0)$ only distributes 180 of the 300 and gives C nothing despite C's reliable $+30$. We need a rule that **averages marginal contributions over all ways a team can**

The coalition lattice (native diagram)



Takeaway

Every edge is a marginal contribution. Player A adds $+120$ when joining the empty room but $+210$ when joining B ($270 - 60$): contributions **depend on who is already there**. The Shapley value will average over *all* such edges, correctly weighted.

Exercise A2.2 — Fair division in the consultancy game [Math]

Exercise

Use the consultancy game ($v = 0, 120, 60, 0, 270, 150, 90, 300$ in the table order above). A player's Shapley value averages their marginal contribution over the six possible **arrival orders** $ABC, ACB, BAC, BCA, CAB, CBA$ (each equally likely).

1. Compute player A's marginal contribution in each of the six orders, and average them to get φ_A.
2. Repeat for B and C (or use any shortcut you can justify).
3. Verify **efficiency**: $\varphi_A + \varphi_B + \varphi_C = v(\{A, B, C\})$.
4. Is C a *dummy* player? Justify with one coalition.

Solution — Exercise A2.2 (1/2)

Solution — the six arrival orders

Method. In an arrival order, each player's contribution is the jump in v when they walk in. Tabulate all six orders once; then *every* player's Shapley value is a column average of the same table.

Order	A adds	B adds	C adds
$A\,B\,C$	120	$270-120=150$	$300-270=30$
$A\,C\,B$	120	$300-150=150$	$150-120=30$
$B\,A\,C$	$270-60=210$	60	$300-270=30$
$B\,C\,A$	$300-90=210$	60	$90-60=30$
$C\,A\,B$	$150-0=150$	$300-150=150$	0
$C\,B\,A$	$300-90=210$	$90-0=90$	0
mean	$\varphi_A=\frac{1020}{6}=170$	$\varphi_B=\frac{660}{6}=110$	$\varphi_C=\frac{120}{6}=20$

Solution — Exercise A2.2 (2/2)

Solution — efficiency and the dummy check

Part 3 — efficiency.

$$\varphi_A + \varphi_B + \varphi_C = 170 + 110 + 20 = 300 = v(\{A, B, C\}).\checkmark$$

The whole revenue is distributed — nothing left over, nothing invented.

Part 4 — is C a dummy? No. A dummy adds nothing to *any* coalition. But $v(\{A, C\}) - v(\{A\}) = 150 - 120 = 30 \neq 0$: C adds 30 to every non-empty team. C's stand-alone value is 0, yet C's Shapley pay is $20 > 0$ — the value correctly rewards *team* contribution, not solo performance.

Comparison. Equal thirds would overpay C (100 vs. 20); stand-alone pay would underpay C (0) and fail efficiency ($180 \neq 300$). The order-averaged split is the only one we will be able to *axiomatise*.

Common mistake

Computing marginal contributions only for the order A, B, C and calling it a day. A single order is arbitrary — A would get 120 instead of 170 — and the result would depend on the labelling of the players.

The Shapley value: the formula

Rule (Shapley, 1953)

The Shapley value of player j in the game $(\mathcal{P}, v)$ is

$$\varphi_j = \sum_{S \subseteq \mathcal{P} \setminus \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} \Big(v(S \cup \{j\}) - v(S) \Big)$$

Symbol by symbol

- $S \subseteq \mathcal{P} \setminus \{j\}$ — every coalition j could join (all 2^{p-1} subsets not containing j); $v(S \cup \{j\}) - v(S)$ — j's **marginal contribution** on arrival.
- $|S|!$ — arrival orders of S *before* j; $(p - |S| - 1)!$ — orders of the rest *after* j; $p!$ — all orders. So the weight is exactly $\Pr(j$ arrives right after the set $S)$ under a uniformly random order: the formula *is* Exercise A2.2's order-averaging, written without listing orders.
- Equivalent form: weight $= \frac{1}{p}\binom{p-1}{|S|}^{-1}$ — pick j's arrival position first (each size equally likely), then the predecessors uniformly among that size.

Checking the formula on the consultancy game

φ_A=170 — now via the subset formula (p=3)

Weights: $|S| = 0: \ \frac{0!\,2!}{3!} = \frac{1}{3}$; $\quad |S| = 1: \ \frac{1!\,1!}{3!} = \frac{1}{6}$; $\quad |S| = 2: \ \frac{2!\,0!}{3!} = \frac{1}{3}$.

S	weight	$v(S \cup \{A\})$	$v(S)$	term
$\varnothing$	1/3	120	0	40
$\{B\}$	1/6	270	60	35
$\{C\}$	1/6	150	0	25
$\{B, C\}$	1/3	300	90	70
			φ_A	170

Same 170 as the six-order average — with 4 terms instead of 6 orders.

Takeaway

Endpoint coalitions carry the **largest** weight ($\frac{1}{3}$ each): one way to be first or last, $\binom{p-1}{|S|}$ ways to see a middle-sized crowd. For $p = 6$: $\frac{1}{6}, \frac{1}{30}, \frac{1}{60}, \frac{1}{60}, \frac{1}{30}, \frac{1}{6}$ by size — and for each j the weights **sum to 1**: φ_j is a weighted *average* of marginal contributions.

The four axioms — and the uniqueness theorem

Each axiom in one honest sentence

1. **Efficiency:** the shares add up exactly, $\sum_{j=1}^{p} \varphi_j = v(\mathcal{P}) - v(\varnothing)$ — the whole surplus is distributed.
2. **Symmetry:** if $v(S \cup \{i\}) = v(S \cup \{j\})$ for every S containing neither, then $\varphi_i = \varphi_j$ — equal treatment of interchangeable players.
3. **Dummy (null player):** if $v(S \cup \{j\}) = v(S)$ for every S, then $\varphi_j = 0$ — who never adds anything gets nothing.
4. **Additivity:** for two games on the same players, $\varphi_j(v + w) = \varphi_j(v) + \varphi_j(w)$ — playing two games at once changes nothing about either.

Why this matters

Theorem (Shapley, 1953). The Shapley value is the *unique* allocation satisfying all four axioms. So once we accept the four wishes from the motivation slide, there is **no freedom left** — the formula is not one heuristic among many, it is the only answer.

Exercise A2.3 — Weights and the dummy axiom [Math]

Exercise

Work directly with the weight $w(S) = \dfrac{|S|!\,(p-|S|-1)!}{p!}$.

1. For $p = 3$, list $w(S)$ for all four coalitions $S \subseteq \{1,2,3\} \setminus \{j\}$ and show they sum to 1.
2. Prove the **dummy axiom** from the formula: if $v(S \cup \{j\}) = v(S)$ for all S, then $\varphi_j = 0$.
3. Using $w(S) = \frac{1}{p}\binom{p-1}{|S|}^{-1}$, explain in one sentence why the empty and the almost-full coalition carry the largest weight.

Solution — Exercise A2.3

Solution

Part 1 — the four weights for $p = 3$. The subsets of the other two players are $\varnothing$, two singletons, and the pair:

$$w = \frac{0!\,2!}{3!},\ \frac{1!\,1!}{3!},\ \frac{1!\,1!}{3!},\ \frac{2!\,0!}{3!} = \frac{1}{3} + \frac{1}{6} + \frac{1}{6} + \frac{1}{3} = 1.\checkmark$$

Part 2 — dummy. Every term of $\varphi_j = \sum_S w(S)\,\big(v(S \cup \{j\}) - v(S)\big)$ contains the factor $v(S \cup \{j\}) - v(S)$, which is 0 for every S by assumption; a sum of zeros is 0. (Because the weights sum to 1, the same argument shows more: φ_j always lies between j's smallest and largest marginal contribution.)

Part 3 — endpoint weights. Writing $w(S) = \frac{1}{p}\binom{p-1}{|S|}^{-1}$: each *size* $|S|$ is equally likely $(1/p)$, and that probability is shared equally among the $\binom{p-1}{|S|}$ coalitions of that size — sizes 0 and $p-1$ have only *one* coalition each, so each keeps the full $1/p$.

Common mistake

Reading the weight as "all 2^{p-1} coalitions are equally important". They are not: for $p = 6$ a single middle coalition carries weight $\frac{1}{60}$, the empty one $\frac{1}{6}$ — ten times more.

Outline

Features are players: the value function of a prediction

Fix the trained model f and one instance x to explain. Let the **features be the players**; a coalition S means "these feature values are *known* to be x_S".

Rule (SHAP value function, marginal form)

$$v_x(S) \;=\; E\big[f(x_S, X_{\bar{S}})\big] \;-\; E\big[f(X)\big] \;\approx\; \frac{1}{B}\sum_{b=1}^{B} f\big(x_S, x_{\bar{S}}^{(b)}\big) - \varphi_0$$

where $\varphi_0 = E[f(X)]$ and the expectation is estimated on a **background sample** $x^{(1)}, \ldots, x^{(B)}$.

How to read this

- $f(x_S, X_{\bar{S}})$: evaluate the model with the **known** features pinned at x's values and the **unknown** ones drawn from data — an average prediction given what we know.
- $v_x(\varnothing) = 0$ and $v_x(\mathcal{P}) = f(x) - \varphi_0$: knowing nothing gives the average prediction; knowing everything gives $f(x)$.
- **Efficiency** then reads $\sum_i \varphi_i = f(x) - \varphi_0$: the attributions decompose *this*

Marginal or conditional? The one modelling choice that matters

Two ways to "not know" the features outside S:

Marginal (interventional)

$$v_x(S) = E\big[f(x_S, X_{\bar{S}})\big] - \varphi_0$$

Draw $X_{\bar{S}}$ from its own distribution, **ignoring** its correlation with x_S — like an intervention on the model's inputs.

Conditional (observational)

$$v_x(S) = E\big[f(X) \mid X_S = x_S\big] - \varphi_0$$

Draw $X_{\bar{S}}$ from its distribution **given** $X_S = x_S$ — what we would expect from the data.

Each option's honest price

Marginal respects the *model*: unused features are true dummies ($\varphi_j = 0$), but the average runs over feature combinations the data never produces (off-manifold — pitfalls section). **Conditional** respects the *data*: no impossible points, but correlated proxies of a used feature get credit even if the model never touches them — the dummy axiom fails *by design*, and the conditional distributions must themselves be estimated.

Takeaway

This module (like most software defaults) uses the **marginal** form: exact dummies, easy estimation — and we will look its off-manifold problem in the eye.

Our testbed: gradient boosting on `Hitters`

- **Data:** `Hitters` — 322 baseball players (1986/87 season); 59 missing salaries dropped $\Rightarrow n = 263$.
- **Target:** log(`Salary`) (salaries in \$1 000, range 67.5–2 460; the log tames the right tail).
- **Features ($p = 6$):** `Years`, `CHits`, `CRBI`, `Walks`, `Hits`, `PutOuts` — small enough for *exact* $2^6 = 64$-coalition Shapley.
- **Model:** `GradientBoostingRegressor` (sklearn defaults, `random_state=2024`): training $R^2 = 0.965$.
- **Background:** $B = 100$ rows drawn with `default_rng(2024)`; baseline $\varphi_0 = 5.854$ log-dollars $\approx$ \$349k.

The instance we will explain

Row 188: a **24-season veteran** holding the career-hits record — `Years` = 24, `CHits` = 4 256, `CRBI` = 1 314, `Walks` = 30, `Hits` = 52, `PutOuts` = 523. Actual 1987 salary: \$750k. Model: $f(x) = 6.610 \approx$ \$743k.

Run this live in the lab notebook

Section §1–§2 of `advanced_02_shapley_lab.ipynb` loads `Hitters`, fits this exact model and builds the background sample — run it now so the later cells have f, bg and base in memory.

Exercise A2.4 — The duplicated feature [Concept]

Exercise

A bank's dataset accidentally contains `income` twice (`income_a` = `income_b` for every customer). The fitted model happens to split **only** on `income_a`.

1. Under the **marginal** value function, what is $\varphi_{\texttt{income\_b}}$? Which axiom decides it?
2. Under the **conditional** value function, why does `income_b` receive a non-zero share?
3. Which of the two constructions evaluates the model at customer profiles that cannot exist (`income_a` $\neq$ `income_b`)?
4. Give one question for which the marginal answer is the right one, and one for which the conditional answer is.

Solution — Exercise A2.4

Solution

Part 1. $\varphi_{\texttt{income\_b}} = 0$ by the **dummy axiom**: the model never touches `income_b`, so pinning it changes no prediction, so every marginal contribution is 0.

Part 2. Conditioning on $\texttt{income\_b} = x$ forces $\texttt{income\_a} = x$ too (they are identical), so $E[f(X) \mid \texttt{income\_b}]$ moves exactly as much as conditioning on `income_a` — informationally the twins are symmetric, and they share the credit. The dummy axiom fails, *deliberately*.

Part 3. The **marginal** one: computing $v(\{\texttt{income\_a}\})$ it pins `income_a` at the customer's value while drawing `income_b` from the background — creating profiles with two different incomes, which the data manifold does not contain.

Part 4. *"Does the deployed model discriminate on `income_b`?"* — marginal (mechanistic: no, it cannot). *"Does the score reveal / depend on the information in `income_b`?"* — conditional (informational: yes, fully).

Common mistake

Believing one of the two is "the correct Shapley value". They are exact answers to *different questions* — the choice of v is part of the explanation and must be reported with it.

Outline

Python lab — run it live in the notebook

Companion notebook: `advanced_02_shapley_lab.ipynb`

The code in this section is meant to be **demonstrated live**. Open the companion Jupyter notebook (in `Advanced/Lab_Notebooks/`) and run it cell by cell:

- §1, *Data and model* — `Hitters`, log salary, the boosted model;
- §2, *The value function* — background sample, $v_x(S)$, the baseline;
- §3, *Exact Shapley by enumeration* — all 64 coalitions, efficiency check;
- §4, *Monte-Carlo permutation sampling* — convergence to the exact values;
- §5, *Local and global pictures* — waterfall, mean $|\varphi_j|$, dependence;
- §6, *Pitfalls* — correlated features and retrain instability.

Data loads via the `ISLP` package or the bundled CSV files; *Lecture exercises — worked Python solutions* and open exercises end the notebook.

Exact enumeration costs 2^p — count before you compute

The formula for one φ_j touches all 2^{p-1} coalitions; all p features together need every one of the 2^p values $v_x(S)$, each an average over B background rows.

The same count as best-subset selection (Ch. 0b and Ch. 6)

	$p = 6$ (here)	$p = 20$	$p = 50$
coalitions 2^p	64	1 048 576	$\approx 1.1 \times 10^{15}$
model calls at $B = 100$	6 400	$\approx 10^8$	hopeless

Chapter 0b's counting slide and Chapter 6's best-subset explosion are *literally the same* 2^p: "every feature in or out". There we gave up on exhaustive search; here we can afford it only because $p = 6$.

Takeaway

Exact Shapley is a **small-p luxury**: 6 400 predictions per instance at $p = 6$ (about 1.7 million for all 263 players — seconds on a laptop). Beyond $p \approx 15$–20 ($2^{15} = 32\,768$ coalitions) we

Exact Shapley by hand (1): model and value function

```
import itertools, numpy as np, pandas as pd # std stack
from math import factorial
from sklearn.ensemble import GradientBoostingRegressor

df = pd.read_csv("Hitters.csv").dropna().reset_index(drop=True)
feats = ["Years", "CHits", "CRBI", "Walks", "Hits", "PutOuts"]
X = df[feats].to_numpy(float) # (263, 6)
y = np.log(df["Salary"]).to_numpy() # log $1000s
f = GradientBoostingRegressor(random_state=2024).fit(X, y)
print(round(f.score(X, y), 3)) # train R^2: 0.965

rng = np.random.default_rng(2024) # house seed
bg = X[rng.choice(len(X), 100, replace=False)] # background, B=100
base = f.predict(bg).mean() # phi_0 = E[f]: 5.854

def v(S, x): # marginal value function
    Z = bg.copy() # B background rows
    if S: Z[:, list(S)] = x[list(S)] # pin known features to x
    return f.predict(Z).mean() - base # average prediction - phi_0
```

Exact Shapley by hand (2): the sum over all coalitions

```
p = len(feats) # p = 6
w = {s: factorial(s) * factorial(p - s - 1) / factorial(p)
     for s in range(p)} # weight by |S|

def exact_shapley(x): # x: one row, shape (6,)
    phi = np.zeros(p)
    for j in range(p): # one player at a time
        rest = [k for k in range(p) if k != j]
        for r in range(p): # coalition sizes 0..p-1
            for S in itertools.combinations(rest, r):
                phi[j] += w[r] * (v(S + (j,), x) - v(S, x))
    return phi

x_star = X[188] # the 24-season veteran
phi = exact_shapley(x_star)
print(dict(zip(feats, np.round(phi, 4)))) # one number per feature
fx = f.predict(x_star.reshape(1, -1))[0] # f(x*) = 6.6102
print(round(phi.sum(), 4), round(fx - base, 4)) # efficiency check
```

Run this live in the lab notebook

Notebook §3 runs exactly this code — `itertools.combinations` over all 64 coalitions — and times it, so

The efficiency axiom, verified to machine precision

Output of the previous slide

	Years	CHits	CRBI	Walks	Hits	PutOuts
φ_j	-0.5245	$+0.5626$	$+0.5474$	-0.0452	-0.1084	$+0.3244$

Sum of attributions: $\sum_j \varphi_j = 0.7564$. Prediction minus baseline: $f(x^*) - \varphi_0 = 6.6102 - 5.8538 = 0.7564$.

Takeaway

Efficiency holds **exactly**, not approximately: across all 263 players the largest violation of $\sum_j \varphi_j = f(x) - \varphi_0$ is 4.4×10^{-15} — floating-point noise. This is the property no rival attribution method (gradients, single-feature ablations) gives you for free.

In industry — Model risk: the audit test

Validation teams re-run exactly this check on sampled instances: if reported attributions do not reproduce

Monte-Carlo: sample arrival orders instead of enumerating

Algorithm (permutation sampling; Štrumbelj & Kononenko, 2014)

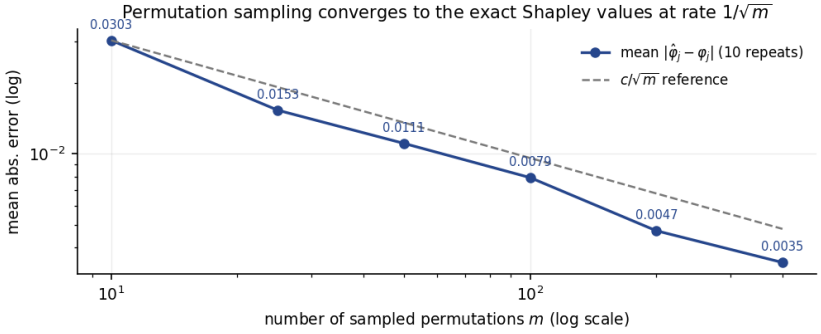
Repeat for $t = 1, \ldots, m$: draw a uniformly random ordering π_t of the p features; walk through it, pinning one feature at a time, and award each feature the change in v_x it causes on arrival. Average:

$$\hat{\varphi}_j = \frac{1}{m} \sum_{t=1}^{m} \Big(v_x\big(\mathrm{pre}_{\pi_t}(j) \cup \{j\}\big) - v_x\big(\mathrm{pre}_{\pi_t}(j)\big) \Big).$$

How to read this

- **Unbiased:** the exact φ_j *is* the expectation of the arrival-order contribution — sampling orders estimates it.
- **Error** $\propto 1/\sqrt{m}$ (a plain Monte-Carlo mean): quadruple the orders, halve the error.
- **Cost:** $m \cdot p$ evaluations of v_x, i.e. $m\,p\,B$ model calls — independent of 2^p. At $p = 6$, $m = 100$: 60 000 calls, — *more* than exact (6 400)! Sampling only pays once 2^p explodes.

Convergence: sampled orders vs. the exact answer



Takeaway

For the veteran instance, $m = 25$ sampled orders already bring the mean absolute error to 0.015 log-dollars ($\sim 2\%$ of the largest $|\varphi_j|$); by $m = 400$ it is 0.0035. The dashed $c/\sqrt{m}$ line confirms the rate: $\times 4$ orders $\Rightarrow$ error halved ($0.0153 \to 0.0079$ from $m = 25$ to 100).

Scaling up in practice: KernelSHAP, TreeSHAP, `shap`

KernelSHAP in one idea (Lundberg & Lee, 2017)

Efficiency suggests a regression: fit $v_x(S) \approx \sum_{j \in S} \varphi_j$ by **weighted least squares** over sampled coalitions S, with the *Shapley kernel* weight $\pi(S) = \frac{p-1}{\binom{p}{|S|}\,|S|\,(p-|S|)}$. With *all* 2^p coalitions this WLS reproduces the exact Shapley values; with a sample it approximates them — efficiently, because each sampled coalition informs all p features at once.

TreeSHAP: exactness for free on tree ensembles

For trees, the expectations $E[f(x_S, X_{\bar{S}})]$ can be pushed *through the tree structure*: exact Shapley values in polynomial time (roughly #trees $\times$ #leaves $\times$ depth^2 per instance) — no 2^p, no sampling noise. This is why boosted trees + SHAP became the industry pairing.

In industry — The production tool

The `shap` package ships all of this (`TreeExplainer`, `KernelExplainer`, the plots). We built everything from scratch so that `shap`'s output is a convenience, not a mystery — it is not installed on the course machines, but one `pip install shap` away in Colab.

Extended Exercise A2.1 — Monte-Carlo Shapley from scratch [Python]

Extended exercise (15 min)

Implement permutation sampling for the veteran instance x^* (row 188) and quantify its accuracy against the exact values from the previous slides.

1. Write `mc_shapley(x, m, rng)`: sample m orderings with `rng.permutation(p)`; for each, walk through the features, awarding each its change in v (start the walk at $v(\varnothing) = 0$).
2. Run it with $m = 100$ and `default_rng(2024)`; tabulate $\hat{\varphi}_j$ against the exact φ_j and report the mean and maximum absolute error.
3. Estimate the error at $m \in \{25, 100, 400\}$ (average over 10 repeats). Does quadrupling m halve the error?
4. Compare total model calls of your $m = 100$ run against exact enumeration at $p = 6$. Which wins here, and when does that flip?

Run this live

The worked solution sits in `advanced_02_shapley_lab.ipynb` under *Lecture exercises — worked Python solutions*: the cell headed *Extended Exercise A2.1 — Monte-Carlo Shapley from scratch [Python]*.

Solution — Extended Exercise A2.1 (1/3)

Solution — code: sampling arrival orders

```
def mc_shapley(x, m, rng): # estimate all p values
    est = np.zeros(p)
    for _ in range(m): # m random arrival orders
        perm = rng.permutation(p) # e.g. [3 0 5 1 4 2]
        S, v_prev = [], 0.0 # empty room: v = 0
        for j in perm: # features walk in
            v_new = v(tuple(S) + (int(j),), x) # value after j joins
            est[j] += v_new - v_prev # j's marginal contribution
            v_prev = v_new # room now includes j
            S.append(int(j))
    return est / m # average over orders

rng = np.random.default_rng(2024) # fresh seeded generator
est = mc_shapley(x_star, 100, rng) # m = 100 orders
```

Method. One pass through an ordering costs p evaluations of v and awards every feature exactly once — so each pass is one telescoping sample of *all* p Shapley values simultaneously.

Solution — Extended Exercise A2.1 (2/3)

Solution — the m=100 run against the exact values

	Years	CHits	CRBI	Walks	Hits	PutOuts
$\hat{\varphi}_j$ (m=100)	−0.5164	+0.5408	+0.5741	−0.0496	−0.1071	+0.3145
φ_j (exact)	−0.5245	+0.5626	+0.5474	−0.0452	−0.1084	+0.3244

Part 2 — errors. Mean absolute error = 0.012, maximum = 0.027 (on `CRBI`) — about 2–5% of the leading attributions: every sign and the full ranking are already correct at $m = 100$.

Part 3 — rate. Averaged over 10 repeats: error 0.0153 at $m = 25$, 0.0079 at $m = 100$, 0.0035 at $m = 400$. Each quadrupling roughly halves the error — the $1/\sqrt{m}$ signature of a Monte-Carlo mean.

Solution — Extended Exercise A2.1 (3/3)

Solution — cost accounting and when sampling wins

Part 4 — model calls.

- **Exact** at $p = 6$: $2^p \cdot B = 64 \times 100 = 6\,400$ calls.
- **MC** at $m = 100$: $m \cdot p \cdot B = 100 \times 6 \times 100 = 60\,000$ calls — *nine times more*, for an approximate answer.

At $p = 6$, exact enumeration wins outright. The flip is fast: exact doubles with every added feature ($2^{20} \cdot B \approx 10^8$ calls at $p = 20$), while the MC budget $m\,p\,B$ grows only *linearly* in p — at $p = 20$ the same $m = 100$ costs 200 000 calls, now 500× cheaper than exact.

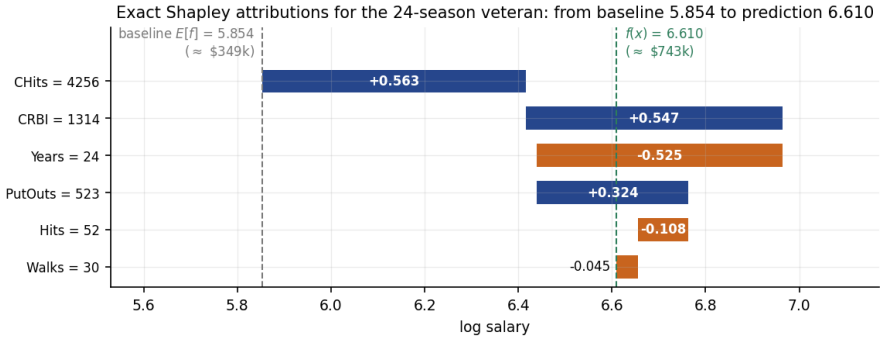
Take-away. Enumerate while you can ($p \lesssim 15$); sample when you must; and if the model is a tree ensemble, TreeSHAP gives exactness at neither price.

Common mistake

Judging convergence from a single run at one m: a lucky draw can look converged while another seed disagrees in the second decimal. Always repeat with several seeds (or track the standard error across orderings) before trusting $\hat{\varphi}_j$.

Outline

Local view: the waterfall of one prediction



Takeaway

Read it left to right: start at the baseline 5.854 ($\approx$ \$349k), add the six attributions in order of size, land exactly on $f(x) = 6.610$ ($\approx$ \$743k). The record career totals push up ($+0.563$, $+0.547$); being 24 seasons old pushes down (-0.525); a weak current season (52 hits) subtracts another -0.108.

Reading attributions on the log scale

Our model predicts *log* salary, so attributions are additive in logs — i.e. **multiplicative** in dollars.

The veteran's salary — rebuilt factor by factor

$$\underbrace{349}_{\text{baseline}} \times \underbrace{e^{+0.563}}_{\texttt{CHits: }\times\mathbf{1.76}} \times \underbrace{e^{+0.547}}_{\texttt{CRBI: }\times\mathbf{1.73}} \times \underbrace{e^{-0.525}}_{\texttt{Years: }\times\mathbf{0.59}} \times \underbrace{e^{+0.324}}_{\texttt{PutOuts}} \times \underbrace{e^{-0.108-0.045}}_{\texttt{Hits, Walks}} \approx 743\ (\$1\,000)$$

All six together: $e^{0.756} = 2.13$ — this player is priced at **2.1 times** the baseline salary, and the waterfall says exactly why.

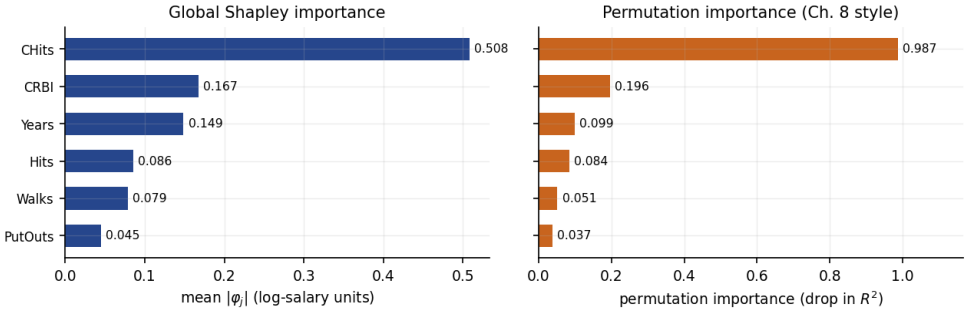
Units checklist before you present

(i) *Which output?* log salary, probability, log-odds, €? (ii) *Which baseline?* $\varphi_0 = 5.854$ is the mean prediction over the $B = 100$ background rows — a different background moves every φ_j. (iii) *Sanity sum:* $\varphi_0 + \sum_j \varphi_j$ must equal $f(x)$.

Run this live in the lab notebook

Notebook §5 rebuilds this waterfall and lets you swap instance and background — watch the baseline and every bar move together.

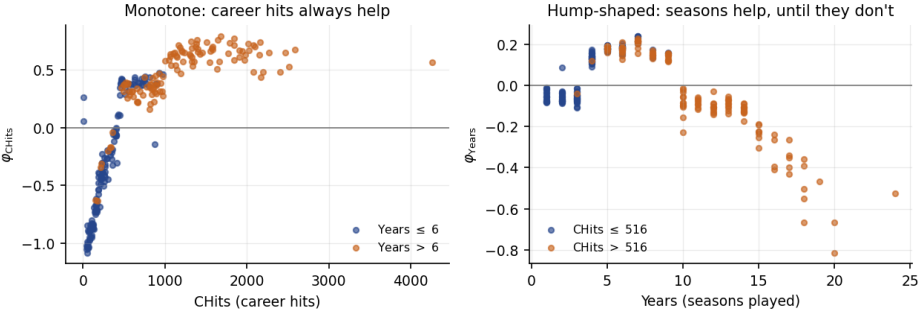
Global view: mean $|\varphi_j|$ vs. permutation importance



Takeaway

Both rankings agree here: `CHits` dominates, then `CRBI` and `Years`. But the *numbers* answer different questions: mean $|\varphi_j|$ (0.508 for `CHits`) is an average **contribution in log-salary units**; permutation importance (0.987) is the **drop in** R^2 when the feature is scrambled (Ch. 8). Under correlation they can diverge: permuting `CHits` also destroys information shared with `CRBI` ($r = 0.947$), inflating its apparent solo importance.

Dependence view: how an attribution varies with its feature



Takeaway

Left: $\varphi_{\texttt{CHits}}$ rises monotonically — mean -0.70 below 250 career hits, $+0.63$ above 1 000: career volume always pays. Right: $\varphi_{\texttt{Years}}$ is **hump-shaped** — $+0.19$ around seasons 5–7, falling to -0.45 beyond 15: *given* career totals, extra seasons mean less production per season. Vertical spread at a fixed x-value (and the colour split) is the fingerprint of **interactions**.

Exercise A2.5 — Explain another player [Python]

Exercise

Row 0 of the cleaned data is a 14-season veteran: `Years` $= 14$, `CHits` $= 835$, `CRBI` $= 414$, `Walks` $= 39$, `Hits` $= 81$, `PutOuts` $= 632$; actual salary \$475k. With the fitted model and `exact_shapley` from the lab:

1. Compute $f(x)$ and the six exact attributions φ_j.
2. Verify efficiency: $\sum_j \varphi_j = f(x) - \varphi_0$.
3. Name the largest positive and the largest negative contribution and restate each as a multiplicative salary factor e^{φ_j}.

Run this live

The worked solution sits in `advanced_02_shapley_lab.ipynb` under *Lecture exercises — worked Python solutions*: the cell headed *Exercise A2.5 — Explain another player [Python]*.

Solution — Exercise A2.5

Solution

Part 1 — the numbers. $f(x) = 6.167$ log-dollars $\approx$ \$477k (actual: \$475k — the model is nearly on the money here).

	CHits	PutOuts	CRBI	Walks	Hits	Years
φ_j	$+0.287$	$+0.176$	$+0.112$	-0.037	-0.100	-0.125

Part 2 — efficiency. $\sum_j \varphi_j = 0.313$ and $f(x) - \varphi_0 = 6.167 - 5.854 = 0.313$: equal to machine precision. ✓

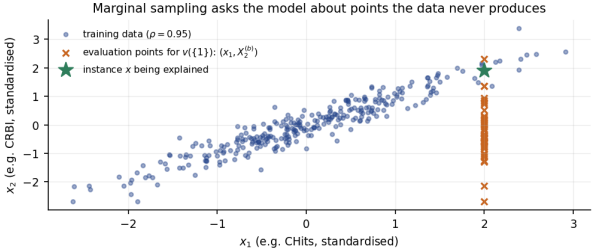
Part 3 — reading. Largest positive: `CHits` $= +0.287$, a factor $e^{0.287} \approx 1.33$ — a solid career record raises the predicted salary by a third. Largest negative: `Years` $= -0.125$, factor $e^{-0.125} \approx 0.88$ — the same late-career discount we saw in the dependence plot (14 seasons sits on its downslope), only milder than the record-holder's -0.525 at 24 seasons.

Common mistake

Comparing φ_j across explanations computed with *different backgrounds*: attributions are shares of $f(x) - \varphi_0$, so changing φ_0 rescales the whole decomposition. One report, one background.

Outline

Pitfall 1: correlated features put the model off-manifold



Takeaway

Marginal sampling pins x_1 and draws x_2 from the background — the orange crosses. With $\rho = 0.95$ most sit **outside** the data ridge: we query the model where it never saw data (in `Hitters`, $r_{\texttt{CHits,CRBI}} = 0.947$, so $v(S)$ averages over impossible careers like 4 256 hits with 3 RBI). The attributions inherit whatever the model *extrapolates* out there.

Common mistake

Trusting fine differences between correlated features' attributions: that split rests exactly on the off-manifold

Pitfall 2: an explanation is not a cause

$\varphi_{\texttt{Years}} = -0.525$ for the veteran. Three readings — only one is licensed:

What the number does and does not say

- ✓ **Model-mechanical:** pinning `Years` $= 24$ (vs. background values) lowers the model's average output by 0.525 — a statement about f and the background, nothing else.
- × **Worldly-causal:** "playing fewer seasons would have raised his salary" — Shapley intervenes on the *model's inputs*, not on the world; no design (randomisation, instruments) supports this.
- × **Recourse:** "to be paid more, reduce `Years`" — feature values cannot be moved independently in reality (`Years` drags `CHits`, $r = 0.898$), and the model would be refitted on any new regime anyway.

Takeaway

Shapley explanations answer "**how did the model use its inputs**", never "what should the

Pitfall 3: attributions move when the model is retrained

Same data and accuracy — different explanations

Two stochastic refits (`subsample=0.7`, seeds 1 and 2) both reach train $R^2 \approx 0.96$. Exact Shapley for the veteran:

	$\varphi_{\texttt{CHits}}$	$\varphi_{\texttt{CRBI}}$	sum of the pair
refit, seed 1	$+0.702$	$+0.521$	$+1.223$
refit, seed 2	$+0.600$	$+0.545$	$+1.146$

$\varphi_{\texttt{CHits}}$ swings by 0.10 ($\sim 15\%$) between two equally accurate models; the **pair's sum** moves far less. Correlated features are interchangeable to the fit — so the fit chooses arbitrarily, and the attribution follows.

In industry — Model risk & regulation: pinning the explanation down

Under SR 11-7-style model governance and the EU AI Act, banks version *everything the explanation depends on*: model artefact, background sample, seed, and `shap` version. A declined customer who reapplies must

Extended Exercise A2.2 — Designing a credit-explanation pipeline [Integrative]

Extended exercise (15 min)

A bank declines consumer loans with a gradient-boosted classifier ($p = 40$ features; `income`, `debt_to_income`, `n_open_lines`, `history_length` form a correlated cluster). Regulation requires the top-3 "principal reasons" on every decline notice. Design the pipeline:

1. Local or global explanations for the notice — and what do you keep the global ones for?
2. Choose and defend a **background**: all applicants, approved applicants, or a single "average customer" row.
3. Exact, permutation-sampling, KernelSHAP or TreeSHAP — pick one and justify with the cost arithmetic of this module.
4. The correlated cluster: one concrete danger for the reason codes, and one mitigation.
5. The applicant replies: "so if I raise my income by €5 000 I will be approved?" — draft the honest two-sentence answer.

Solution — Extended Exercise A2.2 (1/2)

Solution — parts 1–3: the architecture

Part 1 — local for the notice, global for governance. The notice explains *one* decision $\Rightarrow$ local φ_j for that applicant; the top-3 negative attributions map to standardised reason codes. Global mean $|\varphi_j|$ goes to validation: driver stability over time, drift alarms, fair-lending review.

Part 2 — background = approved applicants. The legal question is "why were *you* declined *rather than approved*", so the reference population should be the approved pool: φ_0 becomes the average score of approved customers and negative φ_j's read "what separates you from them". (All-applicants blurs the question; a single average row is not a distribution at all and makes the explanation hostage to one synthetic point.)

Part 3 — TreeSHAP. $p = 40$ kills exact enumeration ($2^{40} \approx 10^{12}$ coalitions); permutation sampling and KernelSHAP cost $m\,p\,B$ model calls *per applicant* with sampling noise on top — awkward when notices must be reproducible. The model is a tree ensemble, so TreeSHAP is exact, fast and deterministic: the only defensible choice here.

Solution — Extended Exercise A2.2 (2/2)

Solution — parts 4–5: correlation and the customer's question

Part 4 — the cluster danger. Credit for the debt burden can split arbitrarily across `income`, `debt_to_income`, `n_open_lines`: after a routine retrain the top-3 list can swap "income too low" for "too many open lines" with *no change in the applicant* — inconsistent notices, regulatory exposure. **Mitigation:** define reason codes on **feature groups** (sum the cluster's φ_j's — efficiency makes group sums well-defined), and freeze model + background + seed per reporting period.

Part 5 — the honest answer. "The reasons describe how our current model weighed your data compared with approved customers; they are not a promise that changing one number changes the decision. A higher income would enter a new application, but the outcome depends on your whole profile at that time."

Common mistake

Publishing reason codes from raw per-feature attributions in a correlated feature space. The maths is exact, but the *split within a cluster* is a modelling artefact — group first, then explain.

Outline

Module A2 in one slide

What we accomplished

Exact, axiom-backed feature attributions for a black-box model — built entirely from scratch.

- Local vs. global explanations, and what an attribution must promise.
- Cooperative games: value functions, marginal contributions, a fully worked three-player division (170/110/20).
- The Shapley value: the $\frac{|S|!(p-|S|-1)!}{p!}$ formula, four axioms, uniqueness.
- Models as games: $v_x(S)$ via background samples; marginal vs. conditional.
- Computation: exact 2^p enumeration ($p = 6$: 64 coalitions), $1/\sqrt{m}$ Monte-Carlo, KernelSHAP, TreeSHAP.
- Reading output: waterfall (349k $\rightarrow$ 743k), mean $|\varphi_j|$ vs. permutation importance, dependence plots.
- Pitfalls: off-manifold sampling, explanation $\neq$ causation, retrain instability.

Key formulas at a glance

Shapley value $\varphi_j = \sum_{S \subseteq \mathcal{P} \setminus \{j\}} \frac{|S|!\,(p-|S|-1)!}{p!} \big(v(S \cup \{j\}) - v(S)\big)$.

Weight identity $\frac{|S|!\,(p-|S|-1)!}{p!} = \frac{1}{p}\binom{p-1}{|S|}^{-1} = \Pr(j \text{ arrives right after } S)$; weights sum to 1.

Order form $\varphi_j = E_\pi\big[v(\mathrm{pre}_\pi(j) \cup \{j\}) - v(\mathrm{pre}_\pi(j))\big]$ — basis of Monte-Carlo sampling, error $\propto 1/\sqrt{m}$.

Model as game $v_x(S) = E\big[f(x_S, X_{\bar{S}})\big] - \varphi_0$ (marginal) vs. $E\big[f(X) \mid X_S = x_S\big] - \varphi_0$ (conditional).

Local accuracy $f(x) = \varphi_0 + \sum_{j=1}^{p} \varphi_j$ (efficiency, applied to the prediction game).

Global importance $I_j = \frac{1}{n}\sum_{i=1}^{n} |\varphi_j^{(i)}|$ (mean absolute attribution).

KernelSHAP kernel $\pi(S) = \frac{p-1}{\binom{p}{|S|}\,|S|\,(p-|S|)}$ (WLS over coalitions recovers φ).

Vocabulary checklist

Term	Meaning
Coalition S	subset of players / features treated as "present"
Value function $v(S)$	worth of a coalition; for models: expected prediction given x_S
Marginal contribution	$v(S \cup \{j\}) - v(S)$: what j adds on arrival
Shapley value φ_j	weighted average of j's marginal contributions
Efficiency / local accuracy	attributions sum exactly to $f(x) - \varphi_0$
Dummy axiom	unused player gets 0
Baseline φ_0	average prediction over the background sample
Background sample	data rows standing in for "unknown" features
Marginal vs. conditional	break vs. respect feature correlations in $v(S)$
Permutation sampling	Monte-Carlo Shapley from random arrival orders
KernelSHAP / TreeSHAP	WLS approximation / exact polynomial-time trees
Waterfall plot	one prediction rebuilt attribution by attribution
Mean $\lvert\varphi_j\rvert$	global importance from local attributions
Off-manifold evaluation	querying the model at impossible feature combinations

Decision rules of thumb

- $p \lesssim 15$ and a cheap model $\Rightarrow$ **enumerate exactly**; verify efficiency to machine precision as a pipeline test.
- Tree ensemble $\Rightarrow$ **TreeSHAP** (exact, deterministic); anything else at large $p \Rightarrow$ KernelSHAP or permutation sampling with repeats and reported Monte-Carlo error.
- Choose the **background to match the question** ("compared to whom?") — and keep it fixed across every explanation you compare.
- Correlated feature clusters $\Rightarrow$ report **group sums** of attributions; distrust the within-cluster split.
- Always state the **output scale** (log, probability, log-odds) and the baseline next to any φ_j.
- Say "the *model* used x_j this way" — never "x_j *caused* the outcome" and never "change x_j to change the decision".

Common pitfalls

Don't

1. Read Shapley values causally or hand them to customers as recourse advice.
2. Compare attributions computed against different backgrounds, model versions or seeds.
3. Trust the credit split *within* a correlated cluster (off-manifold arithmetic decides it).
4. Report sampled Shapley values without a Monte-Carlo error estimate.
5. Confuse mean $|\varphi_j|$ with permutation importance — different units, different question.
6. Treat the explanation as a property of reality — it is a property of *one fitted model* plus *one value-function choice*.

Self-check questions

Can you answer these without the slides?

1. Why is the Shapley weight $|S|!\,(p-|S|-1)!/p!$ exactly the probability that j arrives right after the coalition S?
2. State the four axioms; which one turns Shapley values into an exact decomposition of $f(x)-\varphi_0$?
3. In the consultancy game, why does C earn 20 despite $v(\{C\})=0$?
4. What breaks if you estimate $v_x(S)$ with marginal sampling when $r_{\texttt{CHits},\texttt{CRBI}}=0.947$?
5. You quadruple the number of sampled orderings. What happens to the estimation error, and why?
6. Why did $\varphi_{\texttt{CHits}}$ change from 0.702 to 0.600 across two retrains — and what stayed (almost) put?

Takeaway

If question 4 or 6 feels shaky, re-read the Pitfalls section — that is where explanation projects

Appendix — what is in here, and why

Nothing in the appendix is needed to follow the module: each slide is a formal derivation, a longer worked exercise, or a side topic. Read it when you want the full story, or when a later project sends you back here.

Contents of the appendix

- **The weights, derived** — the counting argument behind $|S|!(p-|S|-1)!/p!$; moved out because the main deck only needs the *reading* of the weight, not its proof.
- **Efficiency, proved** — the telescoping argument; two lines, but a proof nonetheless.
- **KernelSHAP as weighted regression** — the algebra sketch behind the one-slide summary in the main deck.
- **Estimating conditional value functions** — why the "respect-the-data" option is genuinely hard; for readers heading into research or high-stakes deployment.
- **Drill exercise (glove game)** — one more full Shapley computation for exam practice, with solution.

The weights, derived from random arrival orders

Claim

Under a uniformly random ordering π of $\mathcal{P}$,

$$\Pr\big(\mathrm{pre}_\pi(j) = S\big) \;=\; \frac{|S|!\,(p-|S|-1)!}{p!} \qquad \text{for any } S \subseteq \mathcal{P}\setminus\{j\}.$$

The counting

An ordering with $\mathrm{pre}_\pi(j) = S$ is built by: arranging the $|S|$ members of S before j ($|S|!$ ways), then j (fixed), then the remaining $p-|S|-1$ players ($(p-|S|-1)!$ ways). Total favourable orderings: $|S|!\,(p-|S|-1)!$ out of $p!$ equally likely ones. ■

Takeaway

Hence $\varphi_j = \sum_S \Pr(\mathrm{pre}_\pi(j)=S)\,\big(v(S\cup\{j\})-v(S)\big) = E_\pi\big[j\text{'s arrival contribution}\big]$: the subset formula and the order-average are the *same object*, which is why permutation sampling is unbiased.

Efficiency, proved in two lines

Claim

$\sum_{j=1}^{p} \varphi_j = v(\mathcal{P}) - v(\varnothing)$.

Telescoping

Fix any ordering $\pi = (j_1, \ldots, j_p)$ and sum the arrival contributions *along this one ordering*:

$$\sum_{k=1}^{p} \big[v(\{j_1, \ldots, j_k\}) - v(\{j_1, \ldots, j_{k-1}\}) \big] = v(\mathcal{P}) - v(\varnothing),$$

because every intermediate $v(\cdot)$ appears once with $+$ and once with $-$. The sum is the same for *every* ordering, and averaging over π (which is what $\sum_j \varphi_j$ does, by the previous slide) preserves it. ■

Takeaway

Symmetry and dummy are equally quick from the order view; only *uniqueness* needs real work — see Shapley (1953).

KernelSHAP as a weighted regression (sketch)

Represent a coalition S by $z \in \{0,1\}^p$ and posit the **additive surrogate** $g(z) = \varphi_0 + \sum_j \varphi_j z_j$.

The estimator

$$\hat{\varphi} = \arg\min_{\varphi} \sum_{z} \pi(z) \left(v_x(z) - g(z)\right)^2, \qquad \pi(z) = \frac{p-1}{\binom{p}{|z|}\,|z|\,(p-|z|)},$$

with $g(0) = 0$ and $g(1) = f(x) - \varphi_0$ imposed exactly (the kernel is $+\infty$ there, so they act as constraints).

How to read this

- Lundberg & Lee (2017) show: with *all* 2^p coalitions, the WLS solution equals the Shapley values — the kernel is reverse-engineered so that this holds.
- In practice one samples coalitions $\propto \pi(z)$; each sampled $v_x(z)$ informs *all* p coefficients simultaneously — the efficiency advantage over order sampling.
- Caveat: sampled KernelSHAP is still Monte-Carlo — report its error, as with permutation sampling.

Estimating conditional value functions is genuinely hard

The conditional choice $v_x(S) = E[f(X) \mid X_S = x_S] - \varphi_0$ needs the conditional law of $X_{\bar{S}}$ given $X_S = x_S$ — for *every* S.

The estimation menu (Aas et al. 2021)

- **Empirical / kernel:** reweight background rows by closeness to x_S — struggles beyond a few dimensions.
- **Parametric:** assume Gaussian (or copula) features, condition in closed form — clean, but a real assumption about the data.
- **Generative / surrogate:** train a model of $X_{\bar{S}} \mid X_S$ — now the *explanation* has its own model risk.
- **Path-dependent TreeSHAP:** approximates conditioning with the tree's split cover statistics — fast, but neither cleanly marginal nor cleanly conditional.

Takeaway

This is why the marginal form is the pragmatic default: its estimator is a plain average. The price is Pitfall 1 — choose which price you pay, and **say so in the report**.

Exercise A2.6 — The glove game (drill) [Math]

Exercise

Players 1 and 2 each own a **left** glove; player 3 owns a **right** glove. A coalition earns 1 if it holds at least one complete pair (a left *and* a right glove), else 0: $v(S) = 1$ iff $3 \in S$ and $S \cap \{1, 2\} \neq \varnothing$.

1. Write out $v(S)$ for all eight coalitions.
2. Compute φ_3 with the subset formula ($p = 3$ weights $\frac{1}{3}, \frac{1}{6}, \frac{1}{6}, \frac{1}{3}$).
3. Deduce φ_1 and φ_2 *without* further computation, naming the axioms you use.

Solution — Exercise A2.6

Solution

Part 1. $v = 0$ for $\varnothing, \{1\}, \{2\}, \{3\}, \{1,2\}$; $v = 1$ for $\{1,3\}, \{2,3\}, \{1,2,3\}$.

Part 2 — φ_3 via the four coalitions $S \subseteq \{1,2\}$:

$$\varphi_3 = \tfrac{1}{3}\underbrace{(0-0)}_{S=\varnothing} + \tfrac{1}{6}\underbrace{(1-0)}_{S=\{1\}} + \tfrac{1}{6}\underbrace{(1-0)}_{S=\{2\}} + \tfrac{1}{3}\underbrace{(1-0)}_{S=\{1,2\}} = \tfrac{1}{6} + \tfrac{1}{6} + \tfrac{1}{3} = \tfrac{2}{3}.$$

Part 3. Players 1 and 2 are interchangeable in v ($v(S \cup \{1\}) = v(S \cup \{2\})$ for every S avoiding both), so **symmetry** gives $\varphi_1 = \varphi_2$; **efficiency** gives $\varphi_1 + \varphi_2 + \varphi_3 = v(\mathcal{P}) = 1$, hence $\varphi_1 = \varphi_2 = \tfrac{1}{2}\big(1 - \tfrac{2}{3}\big) = \tfrac{1}{6}$.

Take-away. Scarcity is worth money: the single right glove earns $\tfrac{2}{3}$, four times each plentiful left glove — the value prices *replaceability*, exactly what it will later do with correlated features.

Common mistake

Splitting 1 as $\tfrac{1}{3}$ each "because there are three players": equal split ignores that removing player 1 barely matters ($\{2,3\}$ still earns 1) while removing player 3 zeroes the game.