Explainable Artificial Intelligence (XAI)

XAI- INTRODUCTION

Gianni Franchi

MVA course

Course Structure & Evaluation

Learning Path

- **Diverse Modalities:** We will explore XAI techniques applied across various data types: **Sound, Images, and Text (NLP)**.

Continuous Evaluation: Weekly Summaries

- Each week, **XXX students** will be assigned to write a summary of the course.
- These summaries are **graded**. The best summary from each session will be shared with the entire class.

Final Assessment

- **Final Exam:** An individual marked exam at the end of the term.
- **Group Project:** Work in **groups of 4** on a real-world XAI application.
- **Summary of the class**.

What Is Machine Learning?

Machine Learning (ML) is a collection of methods that allow computers to make and improve predictions or decisions based on data.

Example: Predicting house prices by learning patterns from past sales.

Typical applications include:

- House price estimation
- Product recommendation
- Traffic sign detection
- Credit default prediction
- Fraud detection

Although tasks differ, the underlying ML workflow is similar.

Typical Machine Learning Pipeline

Step 1: Data Collection

- Collect large amounts of data
- Data includes outcomes and predictive features

Step 2: Model Training

- Feed data into an ML algorithm
- Learn a predictive model (e.g., classifier, regressor)

Step 3: Deployment

- Apply the model to new data
- Integrate into real systems (cars, finance, websites)

Strengths and Limitations of Machine Learning

Strengths:

- Faster than humans
- Consistent predictions
- Cheap and scalable replication

Limitations:

- Insights are hidden in complex models
- Understanding decisions becomes difficult

Terminology

Algorithm: A set of rules that transforms inputs into outputs. Analogy: cooking recipe (ingredients $\rightarrow$ food).

Machine Learning: Methods that allow computers to learn from data rather than explicit instructions.

ML represents a shift from:

- **Explicit programming**
- to **Data-driven (indirect) programming**

Illustration

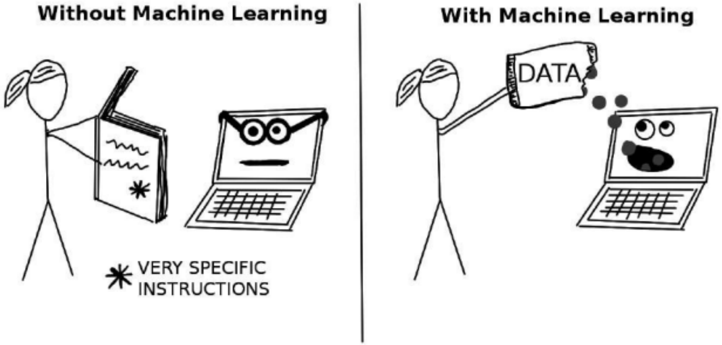


Illustration of classical programming vs machine learning

Machine Learning Models

Machine Learning Model:

- Learned function mapping inputs to predictions
- Examples: linear models, neural networks

Also called:

- Predictor
- Classifier
- Regression model

Notation:

$$\hat{f}(x) \text{ is a model applied on the data } x$$

Black Box vs Interpretable Models

Black Box Model:

- Internal mechanisms are not understandable
- Example: deep neural networks

Interpretable (White Box) Model:

- Internal logic can be understood by humans

Model-agnostic interpretability: Treats any model as a black box, even if it is not.

Definition of Interpretability

There is no formal mathematical definition of interpretability.

Miller (2017):

> *Interpretability is the degree to which a human can understand the cause of a decision.*

Another definition:

> *Interpretability is the degree to which a human can consistently predict the model's output.*

In this course, **interpretable** and **explainable** are used interchangeably.

Why Interpretability Matters

High performance alone is insufficient.

Doshi-Velez & Kim (2017):

> *A single metric such as accuracy is an incomplete description of real-world tasks.*

Interpretability is needed because:

- Problems are often incompletely specified
- The **why** matters as much as the **what**

Human Curiosity and Learning

Humans build mental models of the world.

Unexpected events trigger explanations:

- "Why am I sick?"
- "Why did the computer shut down?"

Without explanations:

- Scientific insights remain hidden
- Learning and trust are limited

Bias, Debugging, and Social Interaction

Bias Detection:

- Models inherit biases from data
- Interpretability helps detect discrimination

Social Interaction:

- Explanations influence beliefs and actions
- Machines must sometimes persuade users

Even low-risk systems benefit from interpretability during development and deployment.

Key Properties Enabled by Interpretability

Interpretability helps assess:

- **Fairness**: absence of discrimination
- **Privacy**: protection of sensitive data
- **Robustness**: stability to small input changes
- **Causality**: learning causal relationships
- **Trust**: user confidence in the system

Taxonomy of Interpretability

Intrinsic vs Post-hoc

- **Intrinsic**: simple, interpretable models (e.g., linear models, small trees)
- **Post-hoc**: explanations applied after training

Global vs Local

- Global: explains overall model behavior
- Local: explains individual predictions

1. Attribution Methods (Feature Importance)

Core Question: Which features or parts of the input mattered most to the prediction?

- **Definition:** These methods assign a numerical score (credit) to each input feature (pixels, words, or tabular columns).
- **Sub-types:**
  - *Perturbation-based:* LIME, SHAP (Sampling).
  - *Gradient-based:* Integrated Gradients, Saliency Maps.
  - *CAM/Grad-CAM:* Class Activation Mapping (specifically for CNNs/Images).
- **Use Case:** Identifying that a model looked at the "ears" to identify a "cat."

2. Concept-Based Explanations

Core Question: Which high-level human ideas/concepts are present in the model's logic?

- **Definition:** Instead of raw features (pixels), these explain the model using human-understandable concepts (e.g., "stripes," "wheels," "beak").
- **Key Techniques:**
  - **CAV (Concept Activation Vectors):** Probes a pre-trained model to see if internal layers "understand" a concept.
  - **CBM (Concept Bottleneck Models):** An intrinsic method where the model is forced to predict concepts before the final label.
- **Use Case:** "The model predicted a 'Zebra' because it detected the concept of 'Stripes'."

3. Example-Based Explanations

Core Question: Which similar or contrasting data points explain the decision?

- **Definition:** Using specific instances from the dataset (real or generated) to justify a classification.
- **Key Techniques:**
  - **Counterfactuals:** "If Feature X had been Y, the outcome would have flipped." (e.g., "If your income was $5k higher, the loan would be approved").
  - **Prototypes/Criticisms:** Showing the "typical" example of a class versus the "outliers" that confuse the model.
- **Use Case:** Debugging by looking at the most similar training images the model was trained on.

4. Rule-Based & Textual Rationalization

Core Question: Can we simplify the logic into rules or natural language?

- **Rule-Based Extraction:**
  - Converting a complex model (like a Random Forest) into a small set of "If-Then" rules.
  - *Example:* IF (Age ¿ 25) AND (Credit ¿ 700) THEN Approve.
- **Textual Rationalization:**
  - Models (often LLMs) that generate a natural language paragraph explaining their reasoning.
  - *Example:* "I classified this as a bird because of the visible feathers and the shape of the wing."

Properties of Individual Explanations

An explanation method is an algorithm that produces explanations.

Key properties include:

- Accuracy
- Fidelity
- Consistency
- Stability
- Comprehensibility

Accuracy and Fidelity

Accuracy:

- How well the explanation predicts true outcomes

Fidelity:

- How well the explanation matches the black-box model
- High fidelity is essential

Some methods only guarantee local fidelity (e.g., Shapley values).

Consistency and Stability

Consistency:

- Similar explanations across different models
- Complicated by the Rashomon Effect

Stability:

- Similar explanations for similar inputs
- High stability is always desirable

Comprehensibility

Comprehensibility:

- How well humans understand explanations
- Depends on the audience

Possible proxies:

- Explanation size
- Number of features or rules
- Human ability to predict model behavior

This is the most critical and difficult property to measure.

Interpretable Models

Linear Regression

Linear Regression: Model Definition

A linear regression model predicts a continuous target variable as a weighted sum of the input features.

For a single instance i, the model is defined as:

$$y^{(i)} = \beta_0 + \sum_{j=1}^{p} \beta_j x_j^{(i)} + \varepsilon^{(i)}$$

- $y^{(i)}$: target value for instance i
- $x_j^{(i)}$: value of feature j
- β_j: learned coefficient (weight) of feature j
- β_0: intercept (bias term)
- $\varepsilon^{(i)}$: error term

Assumptions of Linear Regression

Linear regression relies on several assumptions:

- **Linearity**: The relationship between features and target is linear
- **Independence**: Errors are independent across instances
- **Homoscedasticity**: Constant variance of errors
- **Normality**: Errors ε follow a Gaussian distribution

The Gaussian assumption implies:

- Errors are symmetric
- Small errors occur more frequently than large ones

Estimating the Model Parameters (OLS)

The goal of linear regression is to estimate the coefficients

$$\beta_0, \beta_1, \ldots, \beta_p$$

that best explain the relationship between features and target.

Ordinary Least Squares (OLS) estimates these parameters by minimizing the squared prediction error over the training data:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y^{(i)} - \beta_0 - \sum_{j=1}^{p} \beta_j x_j^{(i)} \right)^2$$

Intuition:

- Each data point produces a prediction error (residual)
- OLS finds coefficients that make all residuals as small as possible
- Squaring penalizes large errors more strongly

How Are the Optimal Parameters Found?

The OLS objective is a convex quadratic function of the parameters.

Key consequences:

- There is a unique global minimum
- No local minima problems

In matrix form, the solution has a closed-form expression:

$$\hat{\boldsymbol{\beta}} = (X^\top X)^{-1} X^\top \mathbf{y}$$

- X: design matrix (rows = instances, columns = features)
- $\mathbf{y}$: vector of target values

We do not need this formula to interpret the model, but it explains why:

- Feature correlations affect coefficient values
- Collinearity increases parameter uncertainty

For details, see Chapter 3.2 of *The Elements of Statistical Learning* (Friedman, Hastie, Tibshirani, 2009)[35].

Why Linear Regression Is Interpretable

Linear regression is inherently interpretable because:

- Each prediction is a sum of feature contributions
- Each coefficient β_j quantifies the effect of feature j
- Contributions are additive and explicit

Explanation source:

- The explanation comes directly from the model equation
- No post-hoc approximation is required

R-squared: What Does It Measure?

The coefficient of determination R^2 measures how well a regression model explains the variability of the target variable.

Intuitively:

- How much of the variation in y is captured by the model?
- How much information do the features provide about the target?

Formally, R^2 compares:

- The error made by the model
- To the natural variability of the data itself

R-squared: Numerator and Denominator

The definition of R^2 is:

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}}$$

Denominator: Total Variability (SST)

$$\text{SST} = \sum_{i=1}^{n} \left(y^{(i)} - \bar{y} \right)^2$$

- Measures how much the target varies around its mean
- Baseline model: always predict the mean $\bar{y}$ ($\bar{y}$ is the mean of the observed data: $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$)

Numerator: Unexplained Variability (SSE)

$$\text{SSE} = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)} \right)^2$$

- Measures the error left after fitting the model

How to Interpret R^2

R^2 quantifies the fraction of variance explained by the model:

$$R^2 = \frac{\text{Explained Variance}}{\text{Total Variance}}$$

- $R^2 = 0$: model is no better than predicting the mean
- $R^2 = 1$: model explains all variability in the data
- $R^2 = 0.7$: 70% of the variance is explained by the features

Why R^2 is useful:

- Model comparison on the same dataset
- Sanity check before interpreting coefficients

Important limitation:

- R^2 always increases when adding features

Adjusted R-squared

R^2 always increases when adding features—even useless ones.

Adjusted R^2 corrects for this:

$$\bar{R}^2 = R^2 - (1 - R^2)\frac{p}{n - p - 1}$$

- p: number of features
- n: number of instances

Models with very low adjusted R^2 should not be interpreted.

Feature Importance via t-statistics

In linear regression, feature importance is assessed by testing whether a coefficient is significantly different from zero.

This is done using the **t-statistic**:

$$t_{\hat{\beta}_j} = \frac{\hat{\beta}_j}{\mathrm{SE}(\hat{\beta}_j)}$$

- $\hat{\beta}_j$: estimated effect of feature j
- $\mathrm{SE}(\hat{\beta}_j)$: uncertainty of this estimate

Interpretation:

- Large $|t|$: strong and reliable effect
- Small $|t|$: effect may be due to noise

What Is $\mathrm{SE}(\hat{\beta}_j)$?

The **standard error** $\mathrm{SE}(\hat{\beta}_j)$ measures how uncertain the estimated coefficient is.

Intuitively:

- It answers: *How much would $\hat{\beta}_j$ change if we collected a new dataset?*

Formally, it comes from the variance of the estimator:

$$\mathrm{SE}(\hat{\beta}_j) = \sqrt{\mathrm{Var}(\hat{\beta}_j)}$$

The variance depends on:

- Noise level in the data
- Number of training instances
- Correlation between features (multicollinearity)

How Is Parameter Variance Computed?

The variance of the estimated coefficients is derived from:

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2 (X^\top X)^{-1}$$

- X: design matrix
- σ^2: estimated noise variance

The standard error is:

$$\mathrm{SE}(\hat{\beta}_j) = \sqrt{\mathrm{Var}(\hat{\beta}_j)}$$

High collinearity $\Rightarrow$ large variance $\Rightarrow$ unreliable explanation.

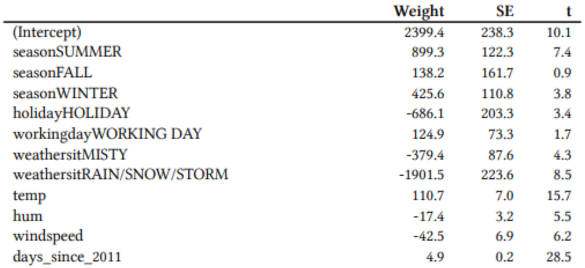
Example: Bicycle Rental Dataset

Linear regression predicts daily bike rentals using:

- Weather conditions
- Calendar variables
- Trend features

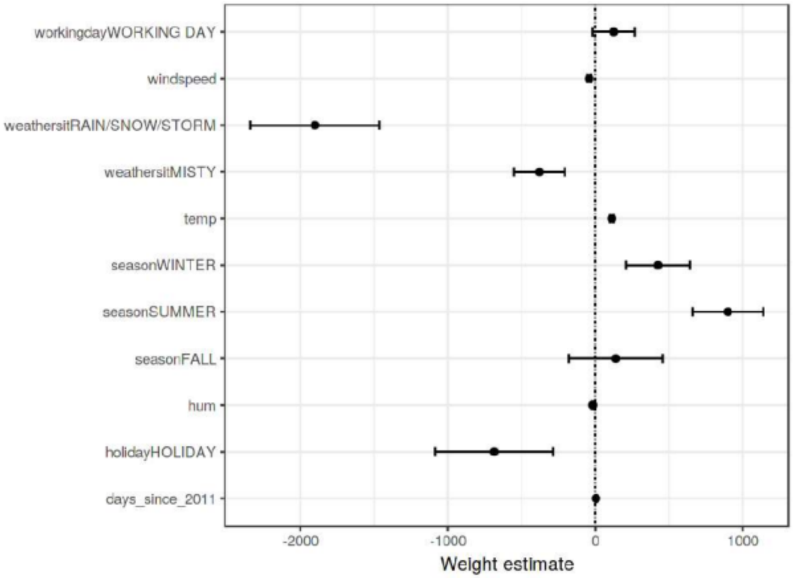
For each feature, we analyze:

- Estimated weight
- Standard error
- Absolute t-statistic



	Weight	SE	t
(Intercept)	2399.4	238.3	10.1
seasonSUMMER	899.3	122.3	7.4
seasonFALL	138.2	161.7	0.9
seasonWINTER	425.6	110.8	3.8
holidayHOLIDAY	-686.1	203.3	3.4
workingdayWORKING DAY	124.9	73.3	1.7
weathersitMISTY	-379.4	87.6	4.3
weathersitRAIN/SNOW/STORM	-1901.5	223.6	8.5
temp	110.7	7.0	15.7
hum	-17.4	3.2	5.5
windspeed	-42.5	6.9	6.2
days_since_2011	4.9	0.2	28.5

Weight Plot



Weights are displayed as points and the 95% confidence intervals as lines.

Interpreting the Weight Plot

- Bad weather has a strong negative effect
- Working day is not statistically significant
- Temperature is significant despite small weight

Limitation:

- Features are on different scales

Solution: standardize features before training.

Effect Plot: Motivation

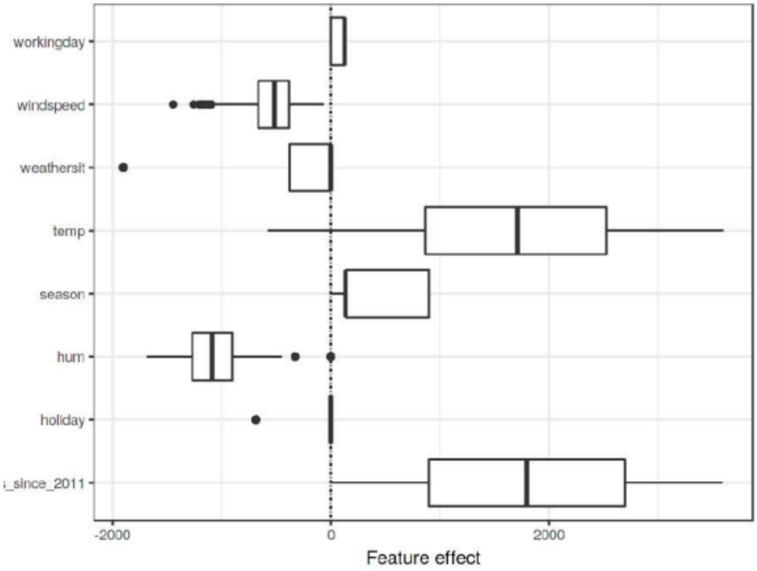
Weights alone are scale-dependent.

The **effect** of feature j for instance i:

$$\text{effect}_j^{(i)} = \hat{\beta}_j x_j^{(i)}$$

Effects represent the actual contribution to the prediction.

Effect Plot Visualization



Interpreting Effect Plots

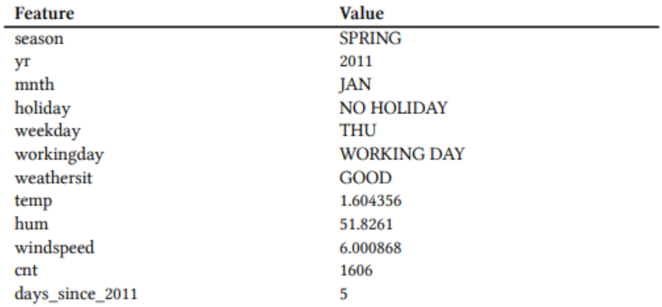
- Temperature and trend dominate predictions
- Trend increases steadily over time
- Negative effects correspond to high negative feature values

Instance-Level Explanation

Question:

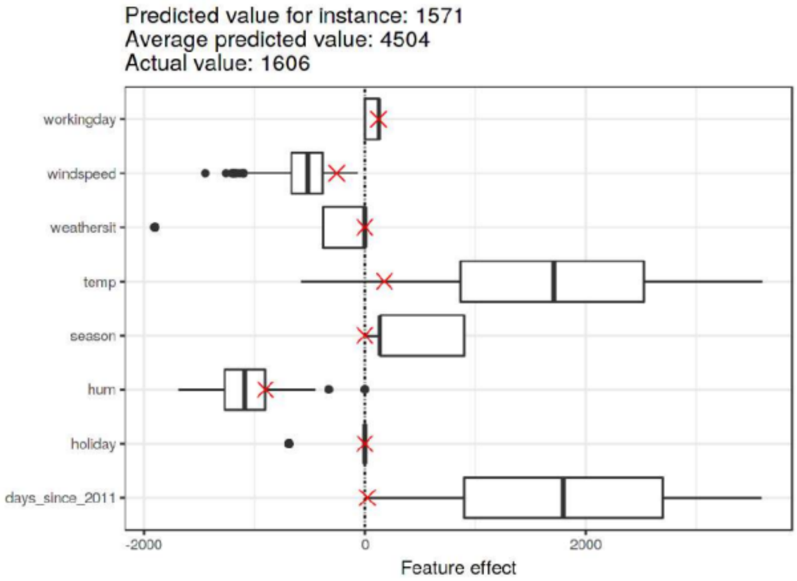
How much did each feature contribute to this prediction?

Compute effects for the selected instance.



Feature	Value
season	SPRING
yr	2011
mnth	JAN
holiday	NO HOLIDAY
weekday	THU
workingday	WORKING DAY
weathersit	GOOD
temp	1.604356
hum	51.8261
windspeed	6.000868
cnt	1606
days_since_2011	5

Comparing Instance Effects



Quality of Linear Explanations

Strengths:

- Truthful (if model assumptions hold)
- Simple and general
- Transparent

Limitations:

- Poor contrastive reference point
- Low selectivity by default

Improving Linear Explanations

Better explanations can be obtained by:

- Mean-centering numerical features
- Effect coding categorical variables
- Using sparse linear models

Linear models remain popular because:

- Linearity simplifies explanations
- Relationships are easy to communicate

Logistic Regression

From Linear Regression to Classification

Linear regression models a continuous outcome as a linear function of the input features:

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \cdots + \beta_p x_p^{(i)}$$

A naive idea for binary classification is to encode the classes as:

$$y \in \{0, 1\}$$

and apply linear regression.

However, this approach is fundamentally flawed:

- The model outputs values in $(-\infty, +\infty)$
- Predictions cannot be interpreted as probabilities
- The model treats class labels as numeric values, not categories

Conclusion: Linear regression is not suitable for probabilistic classification.

Why Linear Regression Fails for Classification

Linear regression fits a hyperplane by minimizing squared errors:

$$\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$$

This leads to:

- Predictions outside the interval $[0, 1]$
- No probabilistic interpretation
- Sensitivity to outliers in label space

Key issue:

- Classification requires modeling probabilities
- Linear regression models numeric targets, not uncertainty

We therefore need a model that:

- Outputs values in $[0, 1]$
- Can be interpreted probabilistically

Logistic Regression: Core Idea

Logistic regression extends linear regression to classification by modeling probabilities.

Instead of predicting $\hat{y}$ directly, we predict:

$$P(y = 1 \mid \mathbf{x})$$

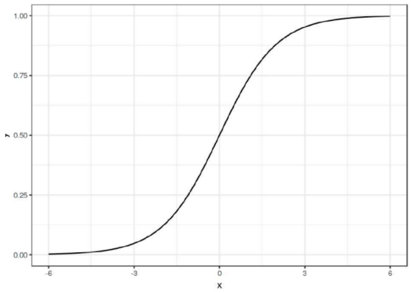
This is achieved by applying a **logistic (sigmoid) function** to a linear combination of the features.

The logistic function is defined as:

$$\sigma(\eta) = \frac{1}{1 + \exp(-\eta)}$$

It maps any real number to the interval $(0, 1)$.

The Logistic Function



The logistic function. It outputs numbers between 0 and 1. At input 0, it outputs 0.5.

Figure: Logistic (sigmoid) function

Properties:

- Smooth and monotonic
- Saturates near 0 and 1
- Naturally interpretable as a probability

Logistic Regression Model Definition

We start from the linear predictor:

$$\eta^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \cdots + \beta_p x_p^{(i)}$$

We then apply the logistic function:

$$P(y^{(i)} = 1) = \frac{1}{1 + \exp\left(-\eta^{(i)}\right)}$$

Explicitly:

$$P(y^{(i)} = 1) = \frac{1}{1 + \exp\left(-(\beta_0 + \beta_1 x_1^{(i)} + \cdots + \beta_p x_p^{(i)})\right)}$$

This guarantees:

$$0 \leq P(y = 1) \leq 1$$

Understanding the Mathematics of Logistic Regression

Logistic regression assumes:

- A linear relationship between features and **log odds**
- Not between features and probabilities directly

The probability is a nonlinear transformation of a linear model.

Explanation comes from:

- The linear structure in log-odds space
- The monotonic mapping from log-odds to probabilities

This preserves interpretability while enabling classification.

From Probabilities to Log-Odds

To interpret the coefficients, we rewrite the model.

Start from:

$$P(y=1) = \frac{1}{1+\exp(-\eta)}$$

Rearranging terms:

$$\frac{P(y=1)}{1-P(y=1)} = \exp(\eta)$$

Taking the logarithm:

$$\log\left(\frac{P(y=1)}{1-P(y=1)}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

This quantity is called the **log odds**.

Odds and Log-Odds

Odds are defined as:

$$\text{odds} = \frac{P(y=1)}{P(y=0)}$$

Log odds:

$$\log(\text{odds}) = \log\left(\frac{P(y=1)}{P(y=0)}\right)$$

Key insight:

- Logistic regression is a linear model in log-odds space
- This is where interpretability comes from

Interpreting a Feature Weight

Consider increasing feature x_j by one unit.

Original odds:

$$\text{odds} = \exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_j x_j + \dots)$$

New odds:

$$\text{odds}_{x_j+1} = \exp(\beta_0 + \cdots + \beta_j (x_j + 1) + \dots)$$

Ratio of odds:

$$\frac{\text{odds}_{x_j+1}}{\text{odds}} = \exp(\beta_j)$$

Interpretation: A one-unit increase in x_j multiplies the odds by $\exp(\beta_j)$.

Interpreting Logistic Regression Coefficients

Numerical feature x_j:

- Increasing x_j by one unit multiplies the odds by $\exp(\beta_j)$

Intercept β_0:

- Represents the odds when all numerical features are zero
- Usually not of practical interest

Why this matters for explanation:

- Effects are multiplicative, not additive
- Coefficients explain changes in odds, not probabilities directly

Decision Trees

What Is a Decision Tree?

A decision tree is a supervised learning model for:

- Classification
- Regression

The model predicts the outcome by recursively splitting the feature space.

Each internal node:

- Tests a feature x_j against a threshold t

Each leaf node:

- Outputs a prediction (class or value)

Key property:

- The model is interpretable by construction

How a Decision Tree Makes Predictions

Given an instance $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_p^{(i)})$:

- Start at the root node
- Evaluate the split condition:

$$x_j^{(i)} \leq t \text{ or } x_j^{(i)} > t$$

- Move to the left or right child node
- Repeat until reaching a leaf

Prediction:

- Classification: majority class in the leaf
- Regression: average target value in the leaf

Explanation comes from:

- The sequence of decisions (path) taken in the tree

Training a Decision Tree: Core Idea

Training a decision tree means learning:

- Which feature x_j to split on
- Which threshold t to use

At each node, the algorithm searches for the split that best separates the data.

Formally, for each candidate split (x_j, t), the data is partitioned into:

$$\mathcal{D}_{\text{left}} = \{i : x_j^{(i)} \leq t\}, \quad \mathcal{D}_{\text{right}} = \{i : x_j^{(i)} > t\}$$

The best split minimizes an impurity measure.

Impurity Measures and Split Criterion

Common impurity measures for classification:

- Gini impurity
- Entropy

Example: Gini impurity at a node:

$$\text{Gini} = 1 - \sum_{k=1}^{K} p_k^2$$

where p_k is the proportion of class k in the node.

The chosen split minimizes the weighted impurity:

$$\frac{|\mathcal{D}_{\text{left}}|}{|\mathcal{D}|} I_{\text{left}} + \frac{|\mathcal{D}_{\text{right}}|}{|\mathcal{D}|} I_{\text{right}}$$

Regression trees:

- Use variance or mean squared error instead

Why Decision Trees Are Interpretable

Decision trees are inherently explainable.

Global explanation:

- The full tree structure shows how features are used
- Feature importance comes from split usage and impurity reduction

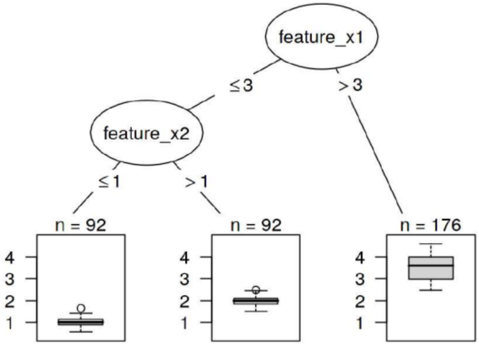
Local explanation:

- A single prediction is explained by its decision path
- Each split corresponds to an interpretable rule

Source of explanation:

- Explicit logical rules learned from data
- No hidden transformations or latent representations

Explanation with a decision tree



Decision tree with artificial data. Instances with a value greater than 3 for feature x1 end up in node 5. All other instances are assigned to node 3 or node 4, depending on whether values of feature x2 exceed 1.

Figure: Decision tree and XAI.

Decision Rules

What Is a Decision Rule?

A **decision rule** is a simple **IF–THEN** statement composed of:

- A **condition** (also called antecedent)
- A **prediction** (also called consequent)

Example (natural language):

IF it rains today AND it is April, THEN it will rain tomorrow.

Key idea:

- If the condition is satisfied, the prediction is applied
- Otherwise, the rule does not apply

Decision Rules as Prediction Models

A predictive model can consist of:

- A single decision rule
- Or a **set of multiple rules**

General structure:

IF (conditions are met) THEN (predict a class or value)

Important:

- In machine learning, rules are **learned automatically**
- Not manually written by a human expert

This distinguishes rule-based ML from traditional rule-based systems.

Why Are Decision Rules Interpretable?

Decision rules are among the **most interpretable models** in machine learning.

Reasons:

- IF–THEN structure resembles natural language
- Mirrors human reasoning
- Each rule can be inspected independently

Interpretability conditions:

- Conditions use intelligible features
- Few feature=value statements per rule
- Limited number of rules

Where does the explanation come from?

- Direct mapping from conditions to prediction
- No hidden transformations or latent representations

Example: House Price Prediction Rule

Consider predicting house value: {low, medium, high}.

Learned decision rule:

$$\text{IF size} > 100 \text{ AND garden} = 1 \text{ THEN value} = \text{high}$$

Interpretation:

- Large houses with a garden tend to have high value
- The model explains its decision using explicit conditions

This explanation is **local and human-readable**.

General Structure of Decision Rules

A decision rule contains:

- At least one condition of the form feature=value or feature>threshold
- Any number of conditions combined using AND

There is **no theoretical upper limit** on the number of conditions.

Exception: Default Rule

- Has no IF-part
- Applies when no other rule applies
- Ensures full coverage of the input space

Model-Agnostic Methods

Model-Agnostic Methods

Model-agnostic explanation methods aim to explain the predictions of a machine learning model *without using any information about its internal structure*.

Key idea:

- Treat the model as a **black box**
- Only require access to:
  - input features x
  - model predictions $\hat{f}(x)$

Why is this important?

- Same explanation method can be used for:
  - linear models
  - tree-based models
  - neural networks
- Enables fair comparison of interpretability across different models

Explanation source:

- Explanations come from *probing model predictions*, not model parameters

Partial Dependence Plots

Partial Dependence Plots (PDP)

Partial Dependence Plots (PDPs) visualize the **average effect** of one or two features on the prediction of a machine learning model.

Introduced by Friedman (2001).

What PDPs answer:

- How does the prediction change on average when a feature value changes?
- Is the relationship linear, monotonic, or non-linear?

PDPs are **global explanations**:

- They summarize model behavior across the entire dataset

Mathematical Definition of PDP

Let $\hat{f}(x)$ be a trained machine learning model.

Split the feature vector into:

- x_S: features of interest
- x_C: all remaining features

The partial dependence function is defined as:

$$\hat{f}_{x_S}(x_S) = \mathbb{E}_{x_C}\left[\hat{f}(x_S, x_C)\right] = \int \hat{f}(x_S, x_C)\, dP(x_C)$$

Interpretation:

- Fix x_S to a specific value
- Average predictions over the distribution of all other features

Explanation source:

- Comes from marginalizing the model prediction over unused features

Practical Estimation of PDP

In practice, the expectation is approximated using the training data (Monte Carlo approximation):

$$\hat{f}_{x_S}(x_S) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}(x_S, x_C^{(i)})$$

Procedure:

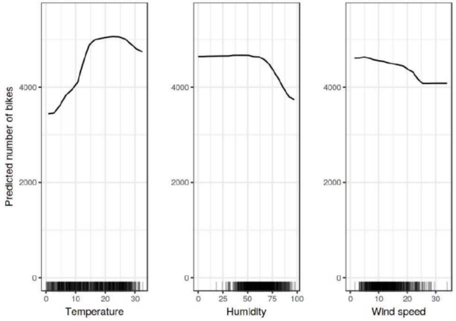
1. Select feature(s) x_S
2. Define a grid of values for x_S
3. For each grid value:
    - Replace x_S in all data points
    - Compute predictions
    - Average predictions

Key assumption:

- Features in x_S are **independent** of features in x_C

PDP Example: Bicycle Rental Prediction

We fit a random forest to predict the number of rented bicycles per day and analyze partial dependence plots.



Interpretation:

- Temperature has the strongest effect
- High humidity reduces bike rentals
- Wind speed shows weaker and uncertain effects

Advantages of PDPs

- Intuitive interpretation:
  - “What happens if we force a feature to a certain value?”
- Easy to explain to non-experts
- Simple to implement
- Provides a causal interpretation *for the model*:
  - We intervene on a feature and observe prediction changes

Important note:

- Causality holds for the model, not necessarily for the real world

Disadvantages of PDPs

- Limited to one or two features
- Strong independence assumption:
  - Violated when features are correlated
- Can create unrealistic data points

Example:

- Height and weight are correlated
- PDP may average over impossible combinations

Consequence:

- Feature effects can be biased

Marginal Plots

Marginal Plots (M-Plots)

Marginal plots (M-plots) address correlated features by averaging predictions **conditionally**.

Interpretation:

- "What does the model predict for instances that actually have this feature value?"

M-plots mix:

- effect of the feature
- effects of correlated features

PDP vs M-Plot: Mathematical Difference

Partial Dependence Plot:

$$\hat{f}_{x_S,\mathrm{PDP}}(x_S) = \mathbb{E}_{X_C}[\hat{f}(x_S, X_C)]$$

Marginal Plot:

$$\hat{f}_{x_S,\mathrm{M}}(x_S) = \mathbb{E}_{X_C|X_S=x_S}[\hat{f}(x_S, X_C)]$$

Key difference:

- PDP averages over the **marginal distribution**
- M-plot averages over the **conditional distribution**

Step-by-Step Computation of an M-Plot

For a feature of interest x_S:

1. Choose a grid of values $v_1, \dots, v_K$ for x_S
2. For each grid value v_k:
   - Select a **local neighborhood** of instances:

     $$\mathcal{N}(v_k) = \{i : |x_S^{(i)} - v_k| \leq \delta\}$$

   - Compute the average prediction:

     $$\hat{f}_{x_S,\mathrm{M}}(v_k) = \frac{1}{|\mathcal{N}(v_k)|} \sum_{i \in \mathcal{N}(v_k)} \hat{f}(x^{(i)})$$

Here, δ controls the neighborhood width (bandwidth).

Explanation source:

- Local averaging approximation of conditional expectations

Global Surrogate Models

Global Surrogate Models

A global surrogate model is an **interpretable model** trained to approximate the predictions of a black-box model.

Let:

- f: black-box model
- g: interpretable surrogate model

Goal:

$$g(x) \approx f(x)$$

Explanation source:

- Interpret g to understand the behavior of f

Training a Surrogate Model

Procedure:

1. Select dataset X
2. Compute black-box predictions $\hat{f}(X)$
3. Choose an interpretable model g
4. Train g on $(X, \hat{f}(X))$
5. Evaluate approximation quality (e.g. R^2)

Important:

- The surrogate never sees true labels
- It learns the *model behavior*, not the data-generating process

Advantages and Disadvantages of Surrogates

Advantages:

- Fully model-agnostic
- Flexible choice of interpretable model
- Easy to communicate
- Quantifiable fidelity using R^2

Disadvantages:

- Explains the model, not the ground truth
- No clear threshold for acceptable R^2

LIME

What Problem Does LIME Solve?

Modern machine learning models are often:

- Highly accurate
- Highly complex (deep networks, ensembles)
- **Not interpretable**

Goal of LIME:

Explain one single prediction of a black-box model in a way that humans can understand.

Key idea:

- Instead of explaining the model *globally*
- Explain it *locally*, around one instance

Explanation source:

- Local approximation theory
- Interpretable surrogate modeling

Why Do We Need a *Local* Model?

A complex model f may behave:

- Highly non-linearly globally
- Approximately linearly in a **small neighborhood**

Analogy:

- The Earth is globally curved
- Locally, the ground is flat

Consequence:

- A simple model can faithfully approximate f
- But only *around a specific point*

This is why LIME is local, not global.

Terminology: The Black-Box Model f

- f is the original trained model
- We can query it with an input x
- It returns a prediction $f(x)$ (e.g., a probability)

Important constraint:

- We do *not* access:
  - Model parameters
  - Gradients
  - Architecture

LIME is model-agnostic.

Original Representation vs Interpretable Representation

Original input:

$$x \in \mathbb{R}^d$$

- Raw pixels, word embeddings, sensor values
- High-dimensional, not human-readable

Interpretable representation:

$$x' \in \{0, 1\}^{d'}$$

- Human-understandable components
- Examples:
  - Image: super-pixels
  - Text: presence of words

Explanation source:

- Interpretability constraints imposed by humans

How Do We Build x'?

Step 1: Decompose the input

- Image $\rightarrow$ super-pixels
- Text $\rightarrow$ words

Step 2: Binary encoding

$$x'_j = \begin{cases} 1 & \text{component } j \text{ is present} \\ 0 & \text{component } j \text{ is hidden} \end{cases}$$

Important:

- x' is only used for explanation
- The black box never sees x'

Why Do We Sample z' Around x'?

We want to understand:

"Which parts of x are responsible for this prediction?"

Idea: Perturbation

- Slightly modify x'
- Observe how the prediction changes

Sampling:

$$z'_i \sim \text{perturbations of } x'$$

- Turn off random interpretable features
- Create local variations

From z' to z: Mapping Back to Input Space

- The black box f expects inputs in original space
- It cannot process z'

Mapping function:

$$z = \text{map}(z')$$

- If $z'_j = 0$, mask the corresponding component
- Example:
  - Image: gray out a super-pixel
  - Text: remove a word

Now we can compute $f(z)$.

Local Weighting with the Similarity Kernel

Not all perturbations are equally important.

Similarity kernel:

$$\pi_x(z) = \exp\left(-\frac{D(x,z)^2}{\sigma^2}\right)$$

- High weight: z close to x
- Low weight: far-away perturbations

Purpose:

- Enforce **local faithfulness**

Explanation source:

- Kernel-weighted local regression

The Surrogate Model g

Surrogate model:

$$g(z') = w^\top z'$$

- Simple
- Interpretable
- Typically linear

Goal:

- Approximate f locally
- Not globally

Explanation source:

- Local linear approximation

LIME Objective Function

The surrogate model is trained by minimizing:

$$\mathcal{L}(f, g, \pi_x) = \sum_{(z,z')\in\mathcal{Z}} \pi_x(z)\big(f(z) - g(z')\big)^2$$

Meaning of each term:

- $f(z)$: black-box prediction
- $g(z')$: surrogate prediction
- $\pi_x(z)$: locality weighting

This is a weighted least-squares problem.

Sparse Explanations via K-Lasso

- Humans cannot interpret hundreds of features
- LIME enforces sparsity

Constraint:

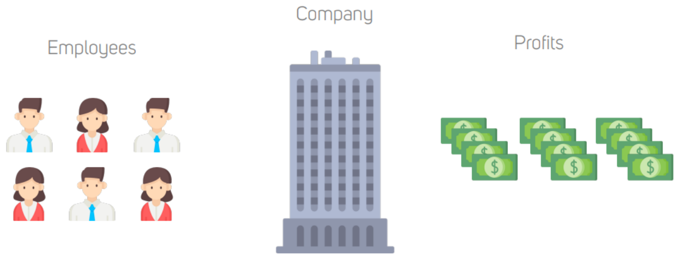
$$\|w\|_0 \leq K$$

Interpretation:

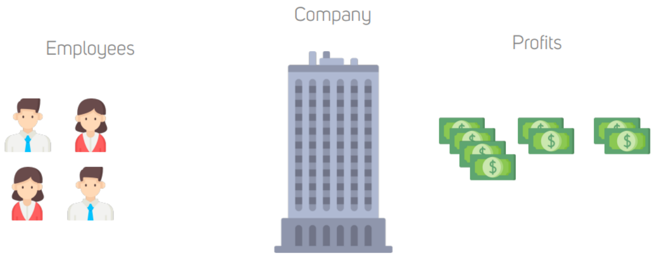
- Only K interpretable features appear
- These are the explanation

SHAP

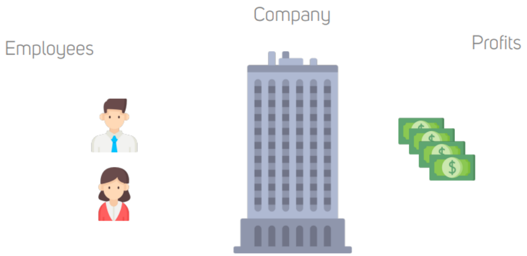
SHAP Illustration (credit Osbert Bastani)



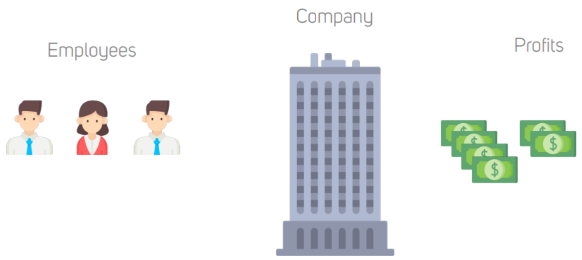
SHAP Illustration (credit Osbert Bastani)



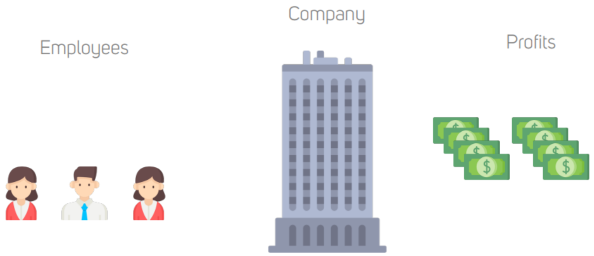
SHAP Illustration (credit Osbert Bastani)



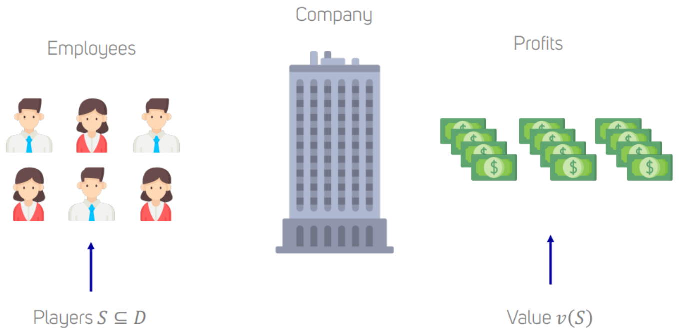
SHAP Illustration (credit Osbert Bastani)



SHAP Illustration (credit Osbert Bastani)



SHAP Illustration (credit Osbert Bastani)



Cooperative Game Notation

- Set of players (features): $D = \{1, \ldots, d\}$
- A coalition: any subset $S \subseteq D$
- A game is defined by a **characteristic function**:

$$v : 2^D \to \mathbb{R}$$

- $v(S)$: value produced by coalition S
- $v(\emptyset)$: baseline value
- $v(D)$: grand coalition value

Key Questions in Game Theory

- Which players are essential?
- How much does each player contribute?
- How should total value be shared fairly?

SHAP answers these questions for ML features.

Shapley Value

- A principled method to allocate credit
- Based on fairness axioms
- Widely used in economics and ML

Lloyd Shapley

- Nobel Memorial Prize in Economics (2012)

Shapley Value Setup

Step-by-step interpretation

- **Input:** one game v
- **Output:** a vector of player credits $\phi(v) = \begin{bmatrix} \phi_1(v) \\ \vdots \\ \phi_d(v) \end{bmatrix} \in \mathbb{R}^d$

So $\phi_i(v) =$ contribution of player i

Shapley Value Example: Setup

Players:

- One owner: o
- n identical employees: $e_1, \dots, e_n$

Player set:

$$D = \{o, e_1, \dots, e_n\}$$

Goal:

- Quantify the contribution of each player
- Using the Shapley value

Coalition Value Function

A cooperative game is defined by a value function:

$$v : 2^D \to \mathbb{R}$$

Definition of the value function:

$$v(S) = \begin{cases} 0 & \text{if } o \notin S \\ (|S| - 1)p & \text{if } o \in S \end{cases}$$

Interpretation:

- If the owner is absent, no production occurs
- If the owner is present, each employee generates profit p
- The owner alone produces no profit

Understanding Marginal Contributions

Key idea of the Shapley value:

- Players enter a coalition one by one
- Contribution is measured when a player joins

For a coalition $S \subset D \setminus \{i\}$, the marginal contribution of player i is: $v(S \cup \{i\}) - v(S)$.

Important observations:

- An employee contributes p only if the owner is already present
- The owner enables employees but does not generate profit alone

For any player i, the Shapley value is defined as:

$$\phi_i(v) = \mathbb{E}_\pi \Big[\underbrace{v(S_i^\pi \cup \{i\}) - v(S_i^\pi)}_{\text{marginal contribution of player } i} \Big]$$

Where: π is a **random permutation** (ordering) of all players and $S_i^\pi \subseteq D$ is the set of players that appear **before** i in ordering π

Why the Shapley Values Take These Values

Shapley value principle:

- Average marginal contributions
- Over all possible player orderings

Employee contribution:

- With probability 1/2, the owner arrives before the employee
- Then the employee adds p
- Otherwise, the contribution is 0

$$\phi_{\text{employee}} = \frac{p}{2}$$

Owner contribution:

- Gains value when employees arrive after him
- On average, enables half of the employees

$$\phi_o = \frac{np}{2}$$

Key Takeaways

- Shapley values fairly distribute total profit:

$$v(D) = np = \phi_o + \sum_{i=1}^{n} \phi_{e_i}$$

- Contributions reflect both:
  - Necessity (owner)
  - Productivity (employees)
- Credit is shared symmetrically and fairly

Explanation source:

- Average marginal contribution over all coalitions
- Core idea behind SHAP in machine learning

Fairness Axioms of the Shapley Value

Setup

- Let $D = \{1, \ldots, d\}$ be the set of players
- Let $v : 2^D \to \mathbb{R}$ be a cooperative game
- Let $\phi(v) = \big(\phi_1(v), \ldots, \phi_d(v)\big)$ be the allocated credits

We want the following fairness properties to hold:

(1) Efficiency

$$\sum_{i \in D} \phi_i(v) = v(D) - v(\emptyset)$$

- The total value of the grand coalition is fully distributed
- No credit is created or lost

(2) Symmetry

$$\forall S \subseteq D \setminus \{i, j\},\; v(S \cup \{i\}) = v(S \cup \{j\}) \;\Rightarrow\; \phi_i(v) = \phi_j(v)$$

- Interchangeable players receive identical credit

Fairness Axioms of the Shapley Value

(3) Null Player

$$\forall S \subseteq D,\ v(S \cup \{i\}) = v(S) \ \Rightarrow\ \phi_i(v) = 0$$

- A player who never adds value gets zero credit

(4) Linearity

$$\phi(c_1 v_1 + c_2 v_2) = c_1 \phi(v_1) + c_2 \phi(v_2), \quad c_1, c_2 \in \mathbb{R}$$

- Credits behave linearly across combined games

Shapley Value Formula

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

Explanation of terms:

- S: coalition without player i
- $v(S \cup \{i\}) - v(S)$: marginal contribution
- Weight: probability of i joining coalition S

Explanation source:

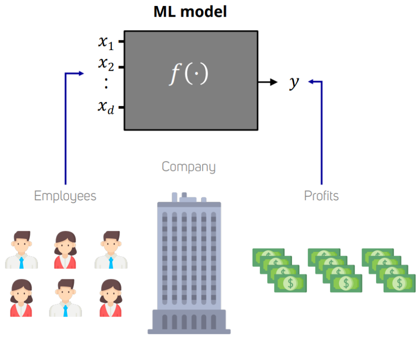
- Average marginal contribution over all permutations

SHAP: SHapley Additive exPlanations (credit Osbert Bastani)

SHAP = SHapley Additive exPlanations **Application to Machine Learning**

- Players → input features
- Game value → model output

Key references: (1)Lipovetsky & Conklin (2001); (2) Strumbelj et al. (2009); (3) Datta et al. (2016).



SHAP as a Removal-Based Explanation

Notation Clarification

- x: instance to explain
- S: subset of features
- x_S: values of features in S
- $x_{\bar{S}}$: missing features

Game definition:

$$v(S) = F(x_S) = \mathbb{E}_{x_{\bar{S}}|x_S}[f(x_S, x_{\bar{S}})] = \sum_{x_{\bar{S}}} f(x_S, x_{\bar{S}})p(x_{\bar{S}}|x_S)$$

Meaning:

- Keep features in S
- Remove others by averaging over their possible values

What is $F(x_S)$?

$$F(x_S) = \mathbb{E}_{x_{\bar{S}}|x_S}[f(x_S, x_{\bar{S}})]$$

Interpretation:

- What would the model predict, if only features in S were known?

Explanation:

- Expected model output for instance x
- Given only features in S

It defines the game value $v(S)$.

Practical Approximation

$$\mathbb{E}_{x_{\bar{S}}|x_S}[f(x_S, x_{\bar{S}})] \approx \mathbb{E}_{x_{\bar{S}}}[f(x_S, x_{\bar{S}})]$$

$$\mathbb{E}_{x_{\bar{S}}}[f(x_S, x_{\bar{S}})] \approx \frac{1}{m} \sum_{i=1}^{m} f(x_S, x_{\bar{S}}^{(i)})$$

- Monte Carlo estimation
- Sample from dataset

Computational Complexity

- Exact Shapley values: $O(2^d)$
- Infeasible for $d > 20$

Solution: Approximation

Algorithm: Calculating Exact Shapley Values

Goal: Calculate the contribution ϕ_i for feature i.

1. **Generate Power Set:** Identify all possible subsets S of the feature set D that do *not* include feature i. There are 2^{d-1} such subsets.
2. **Compute Marginal Contributions:** For each subset S:
   - Calculate the value with the feature: $v(S \cup \{i\})$
   - Calculate the value without the feature: $v(S)$
   - Find the difference: $\Delta_i(S) = v(S \cup \{i\}) - v(S)$
3. **Apply Combinatorial Weights:** Weight each difference by the number of ways that specific subset size can occur:
$$w(|S|) = \frac{|S|!(d - |S| - 1)!}{d!}$$
4. **Aggregate:** Sum the weighted contributions:
$$\phi_i = \sum_{S \subseteq D \setminus \{i\}} w(|S|) \cdot \Delta_i(S)$$

Result: A fair distribution of the prediction "payout" to feature i.

Permutation-Based SHAP

- Randomly sample permutations of features
- Add features one by one
- Track marginal contribution

Approximation:

- Average contributions over permutations

Bibliography:

1 Miller, Tim. "Explanation in artificial intelligence: Insights from the social sciences." Artificial intelligence 267 (2019): 1-38.

2 Doshi-Velez, Finale, and Been Kim. "Towards a rigorous science of interpretable machine learning." arXiv preprint arXiv:1702.08608 (2017).

3 Molnar, Christoph. "Interpretable Machine Learning. Christoph Molnar, 2019."

4 Friedman, Jerome H. "Greedy function approximation: A gradient boosting machine." Annals of statistics (2001): 1189-1232.

EXTRA

Kernel SHAP

- Reformulates SHAP as weighted linear regression
- Uses kernel:

$$\pi(S) = \frac{(d-1)}{\binom{d}{|S|}|S|(d-|S|)}$$

Properties:

- Guarantees Shapley axioms
- Model-agnostic

Kernel SHAP: The Linear Shortcut

- **Problem:** Exact calculation is too slow.
- **Insight:** Since SHAP is an *additive* feature attribution method, can we find the Shapley values using a linear regression?
- **Kernel SHAP** treats the Shapley values as coefficients (ϕ_i) of a local linear surrogate model $g(z')$:

$$g(z') = \phi_0 + \sum_{i=1}^{d} \phi_i z_i'$$

- $z' \in \{0,1\}^d$ is a binary vector representing the presence (1) or absence (0) of a feature.

The Weighted Least Squares Objective

To ensure the coefficients ϕ_i are exactly the Shapley values, Kernel SHAP solves a weighted linear regression:

$$\min_{\phi} \sum_{z' \in \mathcal{Z}} [v(h_x(z')) - g(z')]^2 \cdot \pi_x(z')$$

Where:

- $v(h_x(z'))$: The value function (expected model output) for subset z'.
- $g(z')$: Our linear model.
- $\pi_x(z')$: The **SHAP Kernel** (the weighting function).

The SHAP Kernel $\pi_x(z')$

The mathematical "magic" that makes the linear regression yield Shapley values is this specific weight for each subset size $|z'|$:

$$\pi_x(z') = \frac{(d-1)}{\binom{d}{|z'|}|z'|(d-|z'|)}$$

Why this weight?

- It gives extreme weight to "small" subsets (near 0) and "large" subsets (near d).
- These subsets tell us the most about individual feature effects ($v(\{i\}) - v(\emptyset)$ and $v(D) - v(D \setminus \{i\})$).

Linking back to your slide: $F(x_S)$

As seen in your previous slide, the target for our regression is:

$$v(S) = F(x_S) = \mathbb{E}_{x_{\bar{S}}|x_S}[f(x_S, x_{\bar{S}})]$$

How Kernel SHAP computes this:

1. Sample a binary vector z' (a coalition).
2. Map z' to the original feature space:
   - If $z'_i = 1$, keep x_i from the instance.
   - If $z'_i = 0$, replace x_i with a value from a reference/background dataset (Monte Carlo integration).
3. Pass the resulting vector to the model f to get the "label" for our regression.

Kernel SHAP: Step-by-Step

1. Sample K subsets $z'_k \in \{0, 1\}^d$.
2. For each subset, calculate the model prediction $v(z'_k)$ by simulating the "absence" of features.
3. Calculate the weight $\pi_x(z'_k)$ using the SHAP Kernel formula.
4. Solve the weighted linear regression to find ϕ.
5. Result: The coefficients $\phi_1, \dots, \phi_d$ are the estimated Shapley values.

Summary: Exact vs. Kernel SHAP

Method	**Complexity**	**Approach**
Exact SHAP	$O(2^d)$	Brute force all subsets.
Kernel SHAP	$O(K \cdot d)$	Sample K subsets ($K \ll 2^d$).

- K is a user-defined number of samples (typically a few thousand).
- **Trade-off:** Lower K is faster but increases the variance (error) of the Shapley value estimates.

Why SHAP Satisfies Efficiency and Symmetry

1. **Efficiency (The Summation Logic)**
   - The Shapley value can be viewed as the average of marginal contributions across all $d!$ permutations of players.
   - In any single permutation, the sum of marginal contributions is a **telescoping sum**:

$$(v(\{i_1\}) - v(\emptyset)) + (v(\{i_1, i_2\}) - v(\{i_1\})) + \cdots + (v(D) - v(D \setminus \{i_d\}))$$

   - This always collapses to $v(D) - v(\emptyset)$. Since every permutation sums to the total value, their average (SHAP) must as well.

2. **Symmetry (The Procedural Fairness)**
   - The formula is **permutation-invariant**.
   - If i and j are interchangeable, their marginal contribution to any subset S is identical by definition.
   - Since the formula treats all players identical in the combinatorial weighting, identical contributions result in identical ϕ values.

Why SHAP Satisfies Null Player and Linearity

3. Null Player (The Zero-Contribution Guard)

- Recall the formula: $\phi_i(v) = \sum w(|S|)[v(S \cup \{i\}) - v(S)]$.
- If player i is a "Null Player," the term $[v(S \cup \{i\}) - v(S)]$ is **zero for every single subset** S.
- A sum of zeros is zero. Therefore, $\phi_i(v) = 0$.

4. Linearity (Mathematical Distributivity)

- The Shapley value is a **linear operator**.
- The formula is essentially a weighted sum of the values $v(S)$.
- If we define a new game as $u = c_1 v_1 + c_2 v_2$, we can distribute the summation:

$$\sum w(S)[u(S \cup \{i\}) - u(S)] = c_1 \phi_i(v_1) + c_2 \phi_i(v_2)$$

- This is crucial for ensemble models (like Random Forests), where the SHAP value of the forest is simply the average of the SHAP values of the individual trees.