

# Explainability in AI

Machine Learning & Deep Learning

Jawad ALAOUI

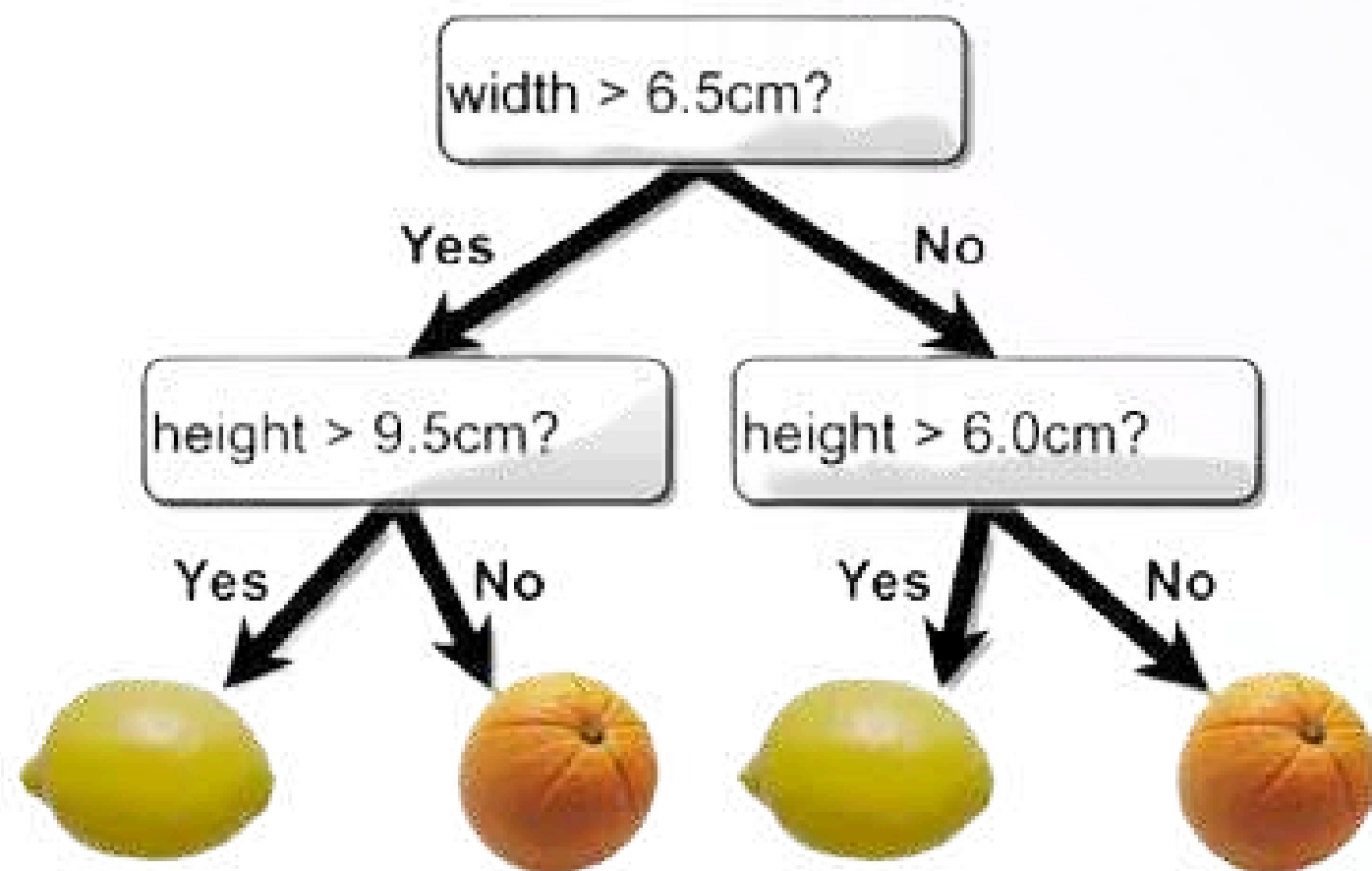


---

# **The Scope of Explainability in AI module**

- Introduction and Motivations
- Taxonomy of Explainability Methods
- Ante Hoc Explainability Methods
- Post-hoc Explainability Methods

# Definition of Explainability in AI



**Explainability (also called interpretability)** is the degree to which a machine learning model's workings and outputs can be understood by humans. In essence, an explainable AI system can articulate why and how a particular decision was made in a way that "makes sense" to a human

**Simple models (like linear models or decision trees)** tend to be inherently explainable, whereas complex models (e.g. deep neural networks) often behave as black boxes that defy easy interpretation.

---

# Why it's important?

## Trust and Transparency

The ability to explain AI decisions is crucial for building trust. When users and stakeholders understand how a model arrives at its predictions, they are more likely to **trust and adopt its recommendations**. Conversely, a lack of transparency can erode confidence – users feel uneasy “in the dark” about an algorithm’s reasoning.

In **high-stakes domains (healthcare, finance, etc.)**, stakeholders demand clear justifications for model decisions before relying on them, as explainability provides reassurance that the AI is making sound and justifiable choices.

---

# Why it's important?

## Ethics and Fairness

Explainability is also essential for identifying and **mitigating bias in AI systems**. Black-box models can inadvertently learn discriminatory patterns from data, leading to unfair or unethical outcomes. Interpretable AI allows developers to detect biased decision rules (e.g. a model overly relying on a sensitive attribute like race, gender or name)

By examining explanations, we can ensure decisions **align with ethical standards** and correct any unjust or harmful behavior in the model, thus promoting fairness and accountability in AI-driven decisions.

---

# Why it's important?

## Regulatory Compliance

In many sectors, laws and regulations now require AI decisions to be explainable. For example, financial and data protection regulations mandate transparency in automated decisions – the **EU's GDPR** grants individuals a “right to an explanation” for algorithmic decisions that affect them

Similarly, laws like the U.S. Equal Credit Opportunity Act demand that lenders provide **specific reasons for loan denials**. Explainable AI facilitates compliance by providing auditable justifications for each outcome, helping organizations meet legal standards and avoid liability.

# Real-World Implications

## Biased AI Model Case Studies

Lack of explainability has led to serious real-world issues, underscoring why it matters.



In the case of the Amazon project, there were a few ways this happened. For example, the tool disadvantaged candidates who went to certain women's colleges presumably not attended by many existing Amazon engineers. It similarly downgraded resumes that included the word "women's" — as in "women's rugby team." And it privileged resumes with the kinds of verbs that men tend to use, like "executed" and "captured."

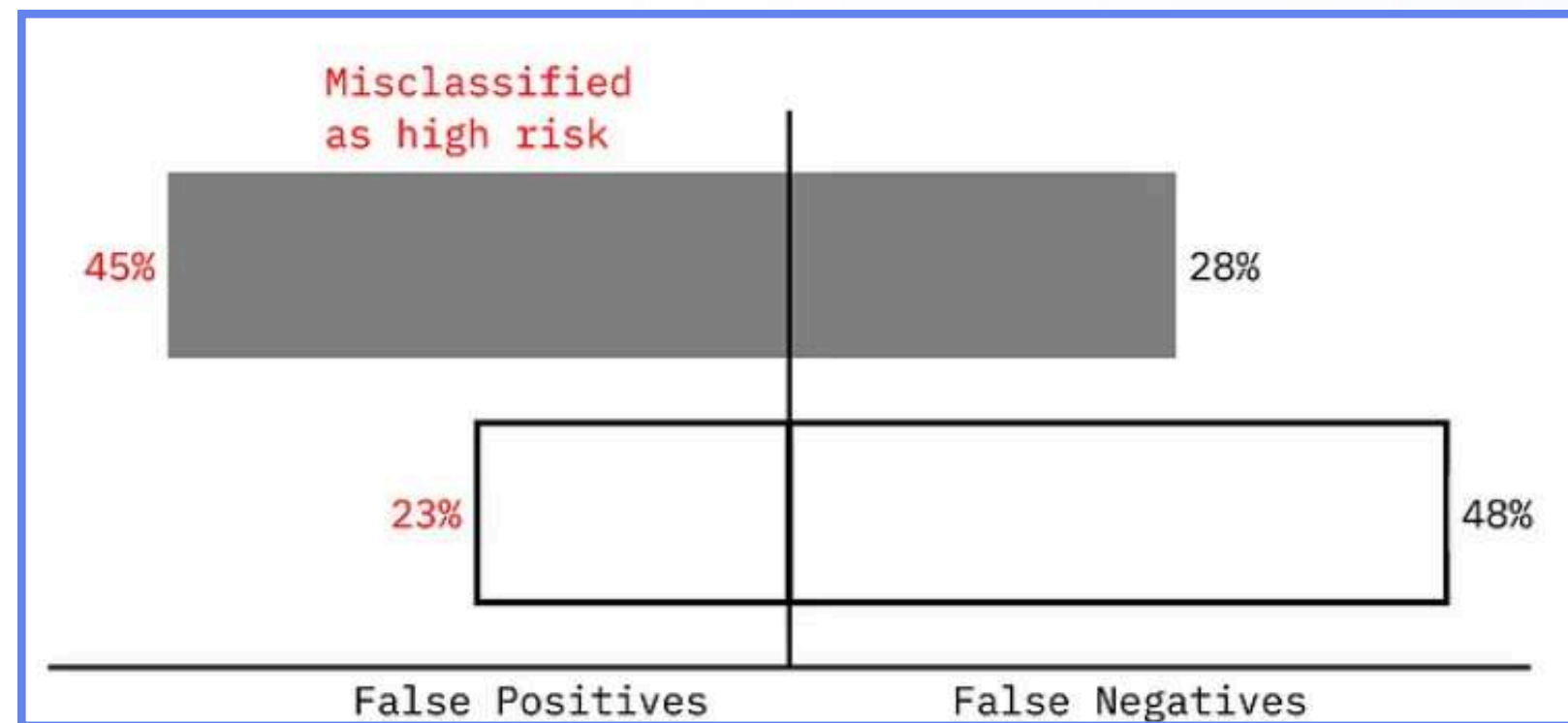
an AI hiring tool developed by Amazon began systematically discriminating against female candidates, downgrading resumes that included the word "women's" – a bias learned from patterns in past hiring data



# Real-World Implications

## Biased AI Model Case Studies

A criminal risk scoring algorithm (COMPAS) was found to disproportionately predict higher recidivism risk for Black defendants compared to white defendants with similar profiles.



True positives & True negatives are cases of the algorithm's predictions being correct. ProPublica found that COMPAS was correct only ~61% of the time.[1] This accuracy rate was similar for both groups.

False positives predict an incorrectly high probability of recidivism. False negatives predict an incorrectly low probability of recidivism. When COMPAS was wrong, it was wrong in different ways for different groups.

Black defendants were more often predicted to reoffend but they didn't. White defendants were more often predicted to not reoffend but they did.

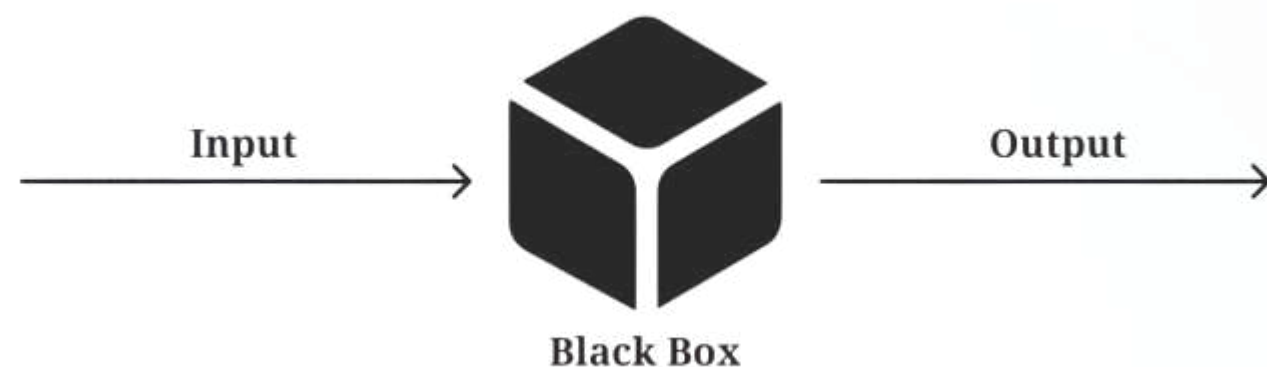
Such cases illustrate that without explainability, AI can “launder” and perpetuate historical biases, with grave consequences for impacted groups.



# Challenges in **Black-Box** AI Models

## Biased AI Model Case Studies

Modern AI models (especially deep learning networks) often operate as black boxes: they have **millions of parameters and complex non-linear interactions** that are not directly interpretable by humans. This opacity means even the engineers building a model may struggle to understand its internal logic or predict why it makes a given prediction.



The absence of insight limits our ability to **debug errors** and trust AI models, especially in safety-critical applications. Without explainability, we risk deploying unpredictable models with faulty reasoning. This drives the need for methods that enhance transparency in AI's decision-making process.

---

# Taxonomy of Explainability Methods

**Intrinsic Explainability:** Intrinsic (built-in) explainability refers to models that are interpretable by design.

- ▶ These models have a **transparent structure that humans can directly follow**, so no external post-processing is needed to understand their decisions.
- ▶ Examples include decision trees (where one can trace the path of decisions), linear/logistic regression (weights indicate feature influence), or rule-based classifiers.

## Interpretability Through Model Constraints

- Achieved by limiting complexity or using human-understandable representations
- Trade-off: Less complex models can be easier to interpret but may have lower accuracy

## Model-Specific Nature

- Interpretable structure depends on the model type
- Example: A small decision tree is intrinsically interpretable, while a neural network typically is not

---

# Taxonomy of Explainability Methods

**Post-hoc Explainability:** Post-hoc methods provide explanations after a model has been trained, without altering the model itself.

- ▶ Treats the model as a black box and infers explanations by **examining inputs and outputs**
- ▶ The goal is to **approximate or interpret what the model is doing** in human-understandable terms, even if the model is complex.

## Approximate Nature of Explanations

- Does not require the original model to be simple
- Explanations are approximations, requiring careful interpretation

## Independant from the explained model

- Applied after model training
- Examples: Surrogate models, Feature importance analysis, Visualization tools (graphs, heatmaps, etc.)

---

# Taxonomy of Explainability Methods

**Global vs. Local Explanations:** Explainability methods can be characterized by the scope of the explanation.

## **Approximate Nature of Explanations**

A global explanation provides a **high-level understanding** of what factors generally drive the model's predictions. For example, global methods might yield a set of decision rules for the entire model or a ranking of feature importances for the model as a whole.

## **Independant from the explained model**

Local predictions explain the model's decision for **specific inputs** by highlighting influential features. Local methods address "Why did the model do X for this case?" while global methods focus on "How does the model generally make decisions?"

Practitioners often use both to understand general patterns and investigate specific cases, particularly outliers or errors.

---

# Taxonomy of Explainability Methods

**Model-Specific vs. Model-Agnostic Methods:** Another axis in explainability methods is whether a technique relies on the internal details of the model

## Model-specific

These methods are **tailored to a particular model** type and take advantage of its internal structure or training process. These methods might not generalize beyond that model family. For example, visualizing attention weights in an NLP model, or using layer-wise relevance propagation in neural networks, are specific to those model architectures

## Model-agnostic

These methods, on the other hand, can be applied to any machine learning model because **they treat the model as a black box**. They only require the ability to query the model with inputs and get outputs, without needing to inspect the model's internals



# Ante Hoc Explainability Methods

## Decision Trees

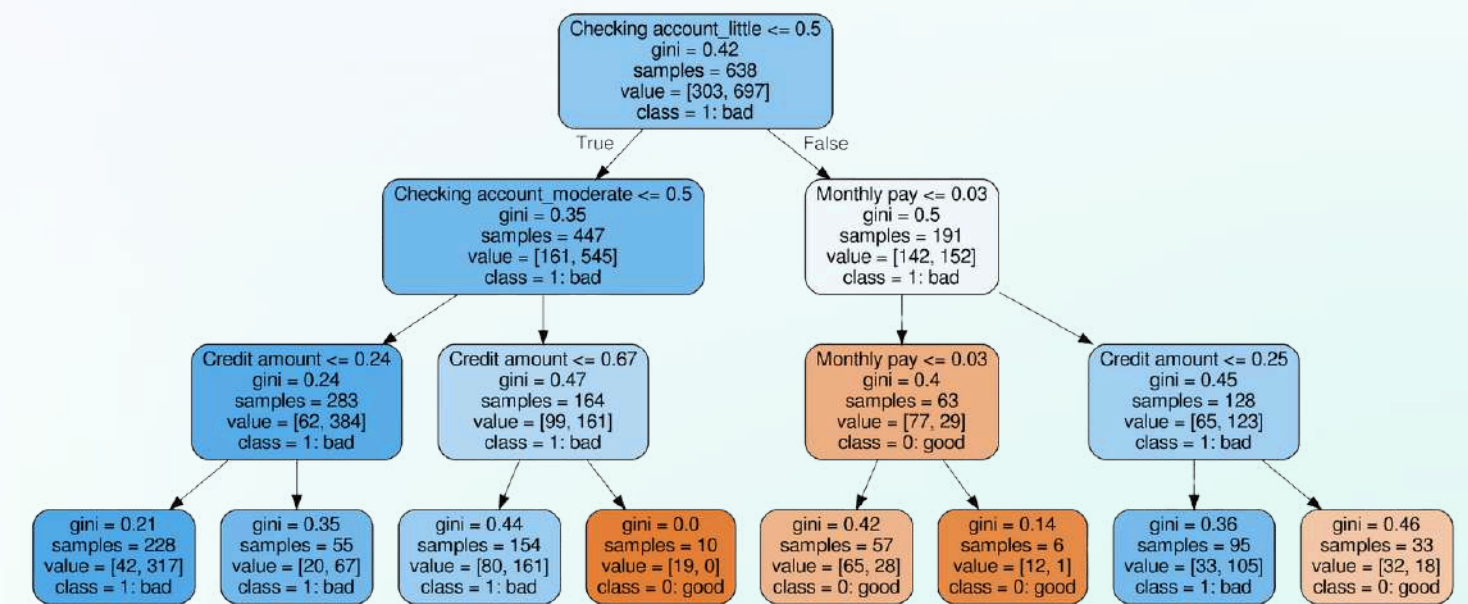
A decision tree is a flowchart-like structure where each internal node represents a test on a feature, each branch represents an outcome of that test, and each leaf node (terminal node) represents a class label (for classification) or a predicted value (for regression).

### Interpretability:

- Easy to understand and interpret after a brief explanation
- Can be displayed graphically for non-experts to interpret
- Each path from root to leaf represents a decision rule.

### Advantages:

- Handles both numerical and categorical data.
- Requires little data preparation; no need for data normalization or dummy variables.
- Reflects the importance of attributes; features on top are the most informative.





# Ante Hoc Explainability Methods

## Generalized linear Models (GLM)

The generalized linear model (GLM) is a generalization of ordinary linear regression, defined by the formula:

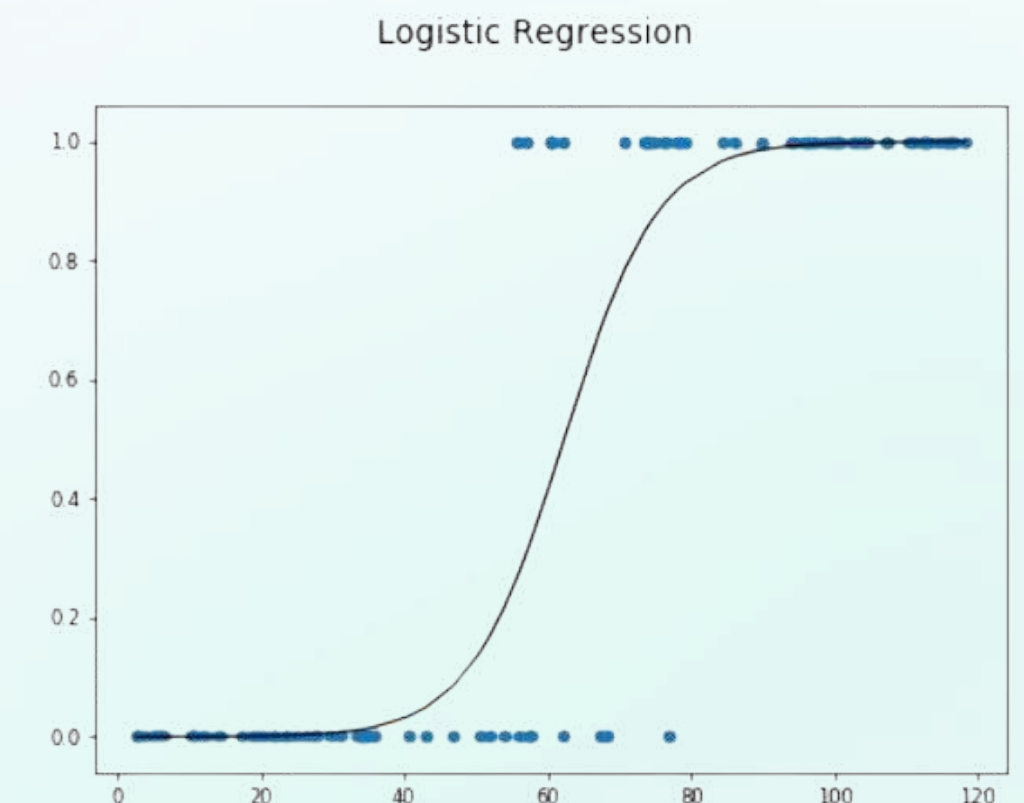
$$E[Y] = g^{-1}(Xa)$$

### Interpretability:

- Coefficient significance is typically assessed using **t-tests** or **z-tests**, where a low p-value indicates a statistically significant predictor.
- Each coefficient represents the change in the transformed mean response per one unit change in the predictor.
- Example: In logistic regression (a GLM with a logit link), a coefficient of 0.5 implies that a one-unit increase in the predictor multiplies the odds by  $\exp(0.5) \approx 1.65$  (i.e., a 65% increase in odds).

### Advantages:

- Flexible framework accommodating various types of response variables.
- Maintains interpretability through model coefficients.



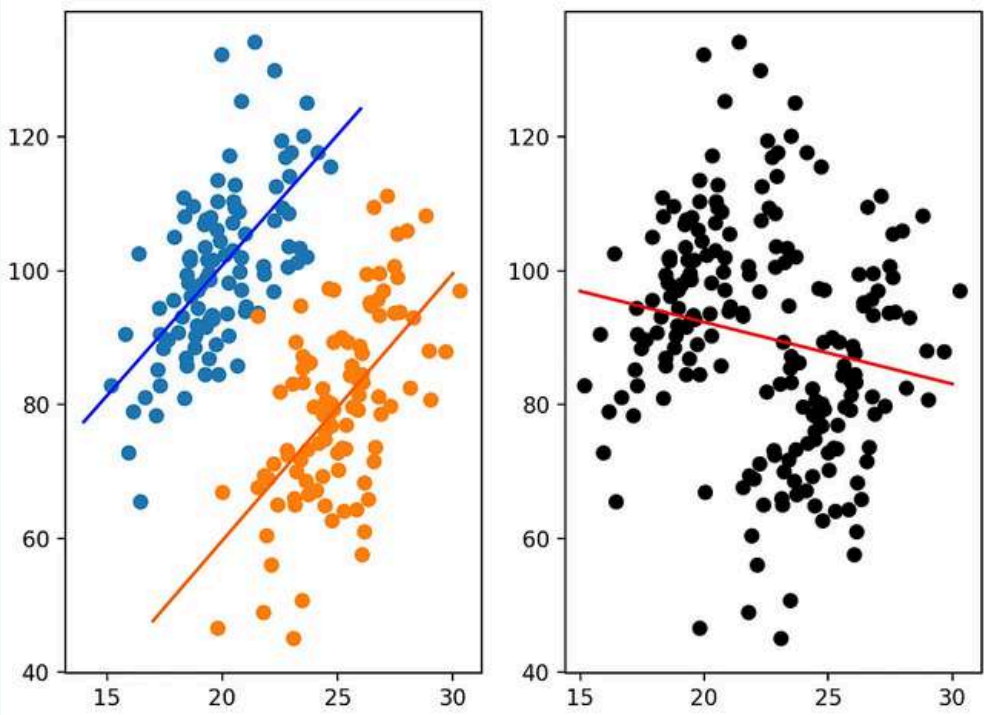
# Ante Hoc Explainability Methods

## Simpson's paradox & confounding variables

Simpson's Paradox refers to a phenomenon in which a trend appears in several different groups of data but disappears or reverses when these groups are combined.

Examples:

	Drug A	Drug B
Effectiveness in male (%)	$\frac{60}{200} \times 100 = 30\%$	$\frac{90}{180} \times 100 = 50\%$
Effectiveness in female (%)	$\frac{240}{300} \times 100 = 80\%$	$\frac{45}{50} \times 100 = 90\%$
Combined (%)	$\frac{300}{500} \times 100 = 60\%$	$\frac{135}{230} \times 100 = 58.69\%$



Confounding variables are factors that, while not the primary focus of a study, can significantly impact how we interpret the relationship between the main variables under investigation. These variables introduce biases or distortions, making it difficult to attribute any observed effects solely to the studied variables.

---

# Post Hoc Explainability Methods

## Global Model-Agnostic Methods

- Permutation Feature Importance (PFI)
- Partial Dependence Plots (PDP)
- Individual Conditional Expectation (ICE )
- Leave One Feature Out (LOFO) Importance
- Feature Interaction
- Surrogate Models

# Post Hoc Explainability Methods

## Feature Importance - Permutation Importance

Permutation importance is a model-agnostic method that assesses the contribution of a feature by measuring the change in model performance when its values are randomly shuffled. This breaks the association between the feature and the target, revealing its importance.

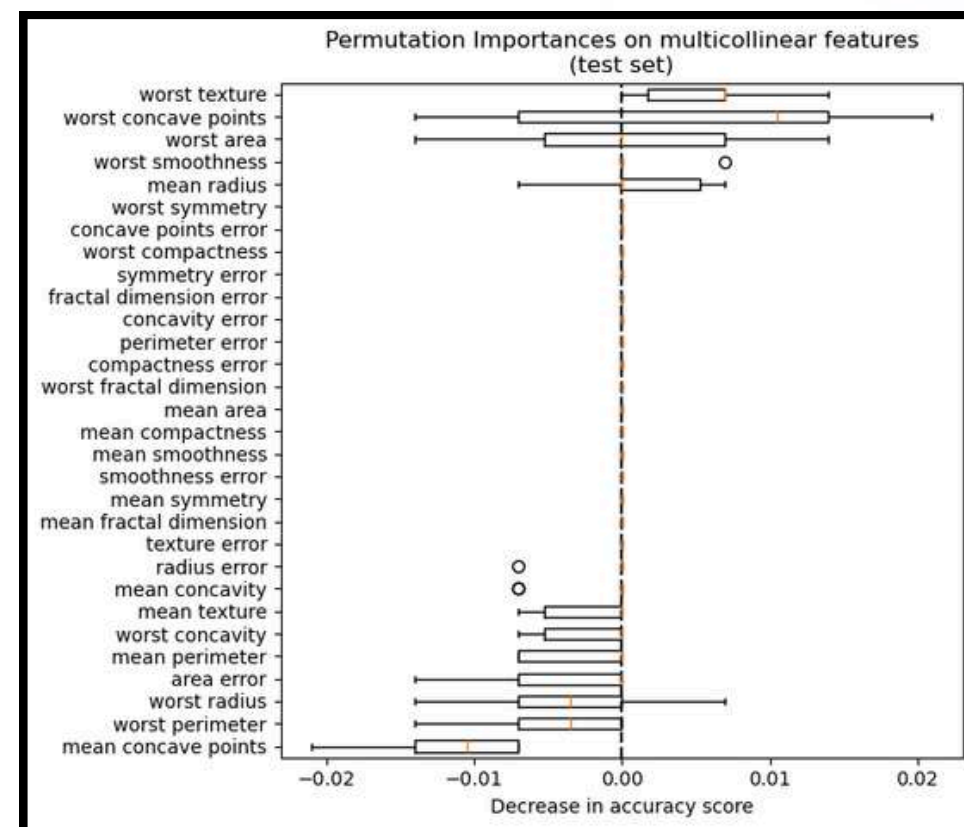
- Inputs: fitted predictive model  $m$ , tabular dataset (training or validation)  $D$ .
- Compute the reference score  $s$  of the model  $m$  on data  $D$  (for instance the accuracy for a classifier or the  $R^2$  for a regressor).
- For each feature  $j$  (column of  $D$ ):
  - For each repetition  $k$  in  $1, \dots, K$ :
    - Randomly shuffle column  $j$  of dataset  $D$  to generate a corrupted version of the data named  $\tilde{D}_{k,j}$ .
    - Compute the score  $s_{k,j}$  of model  $m$  on corrupted data  $\tilde{D}_{k,j}$ .
  - Compute importance  $i_j$  for feature  $f_j$  defined as:

$$i_j = s - \frac{1}{K} \sum_{k=1}^K s_{k,j}$$

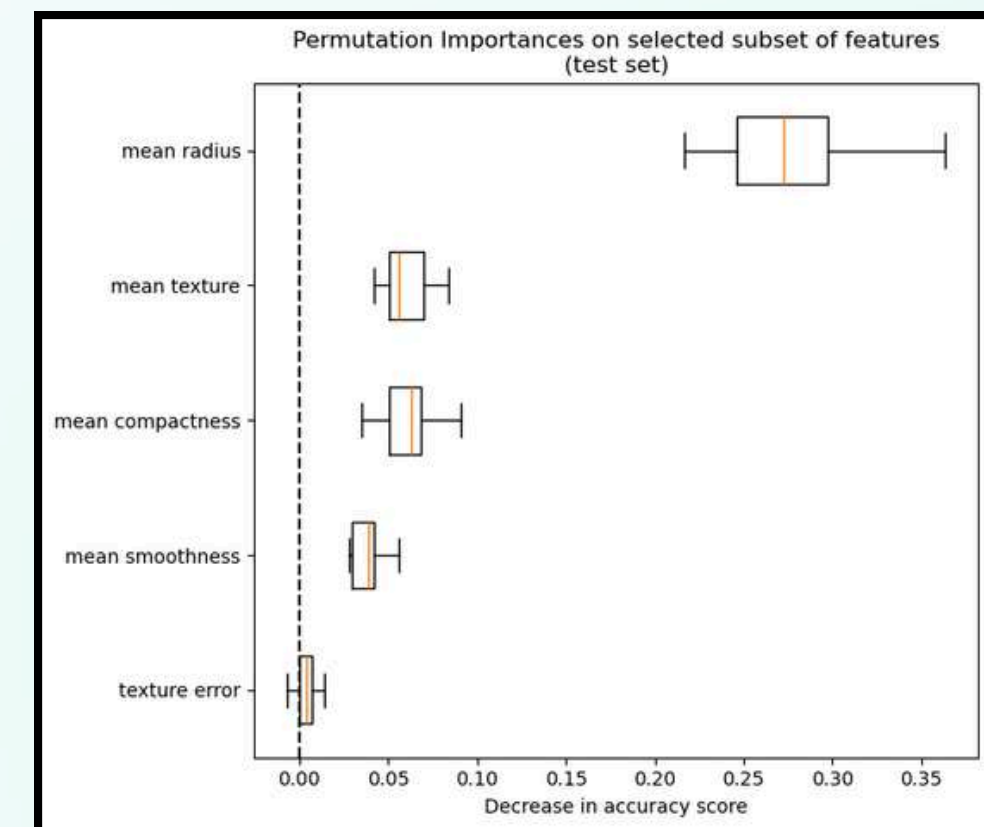
# Post Hoc Explainability Methods

## Feature Importance - Permutation Importance

- Features deemed unimportant in a poor model may be crucial in a good one. Thus, evaluating a model's predictive power with a held-out set or cross-validation is essential before assessing feature importance. Permutation importance indicates a feature's relevance to a specific model, rather than its intrinsic predictive value.
- When two features are correlated and one of the features is permuted, the model still has access to the latter through its correlated feature. This results in a lower reported importance value for both features, though they might actually be important.



after clustering features that are correlated





---

# Post Hoc Explainability Methods

## Feature Importance - Partial Dependence Plots (PDP)

The partial dependence plot (short PDP or PD plot) shows the marginal effect one or two features have on the predicted outcome of a machine learning model. A PDP can show whether the relationship between the target and a feature is linear, monotonic or more complex. For example, when applied to a linear regression model, partial dependence plots always show a linear relationship.

Let  $X_S$  be the set of input features of interest and let  $X_C$  be its complement. PDF at point  $x_S$ :

$$\hat{f}_S(x_S) = \mathbb{E}_{X_C} [\hat{f}(x_S, X_C)] = \int \hat{f}(x_S, X_C) dP(X_C) \quad \hat{f} \text{ the machine learning model}$$

The partial function is estimated by calculating averages in the training data, also known as Monte Carlo method:

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

In this formula,  $x_C^{(i)}$  are actual feature values from the dataset for the features in which we are not interested, and  $n$  is the number of instances in the dataset.



---

# Post Hoc Explainability Methods

## Feature Importance - Individual Conditional Expectation (ICE) plots

Similar to a PDP, an individual conditional expectation (ICE) plot shows the dependence between the target function and an input feature of interest. However, unlike a PDP, which shows the average effect of the input feature, an ICE plot visualizes the dependence of the prediction on a feature for each sample separately with one line per sample.

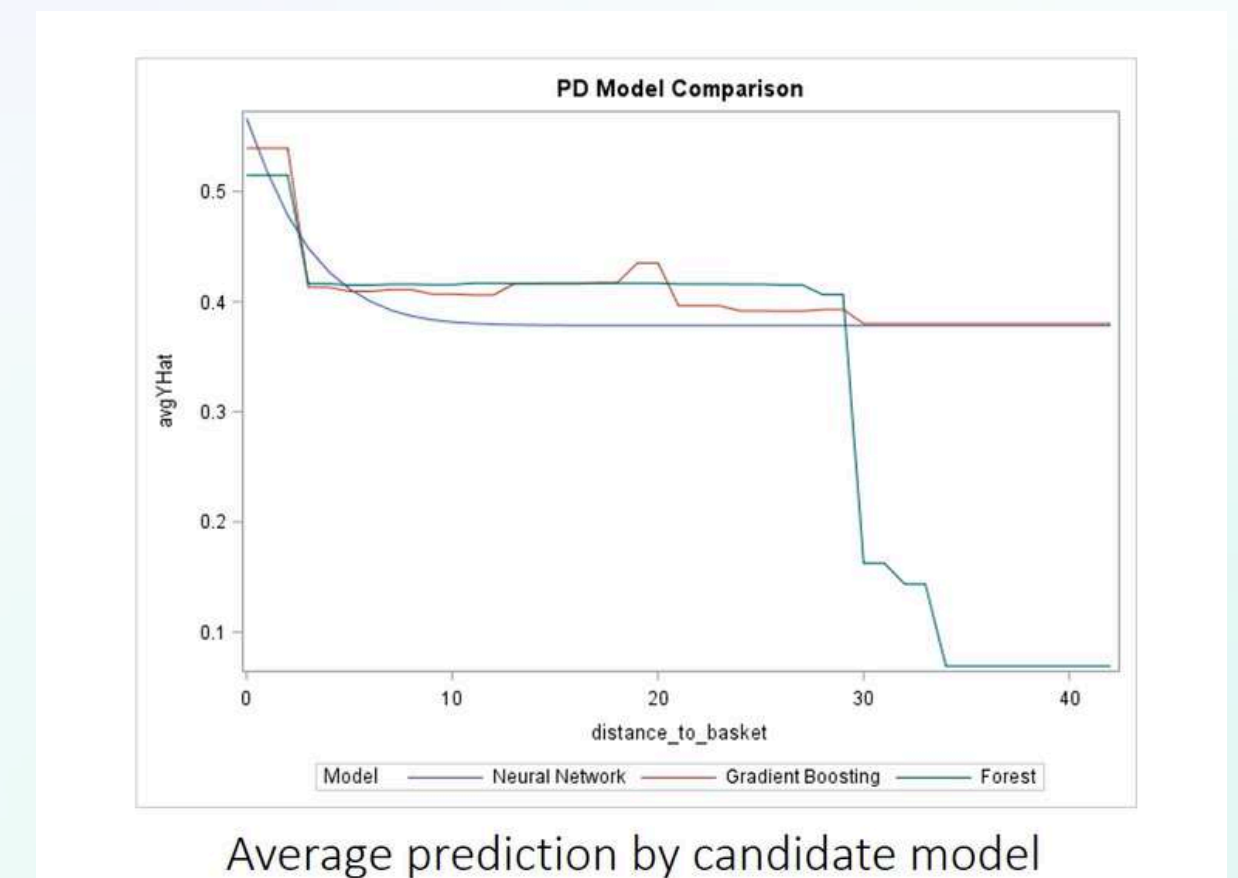
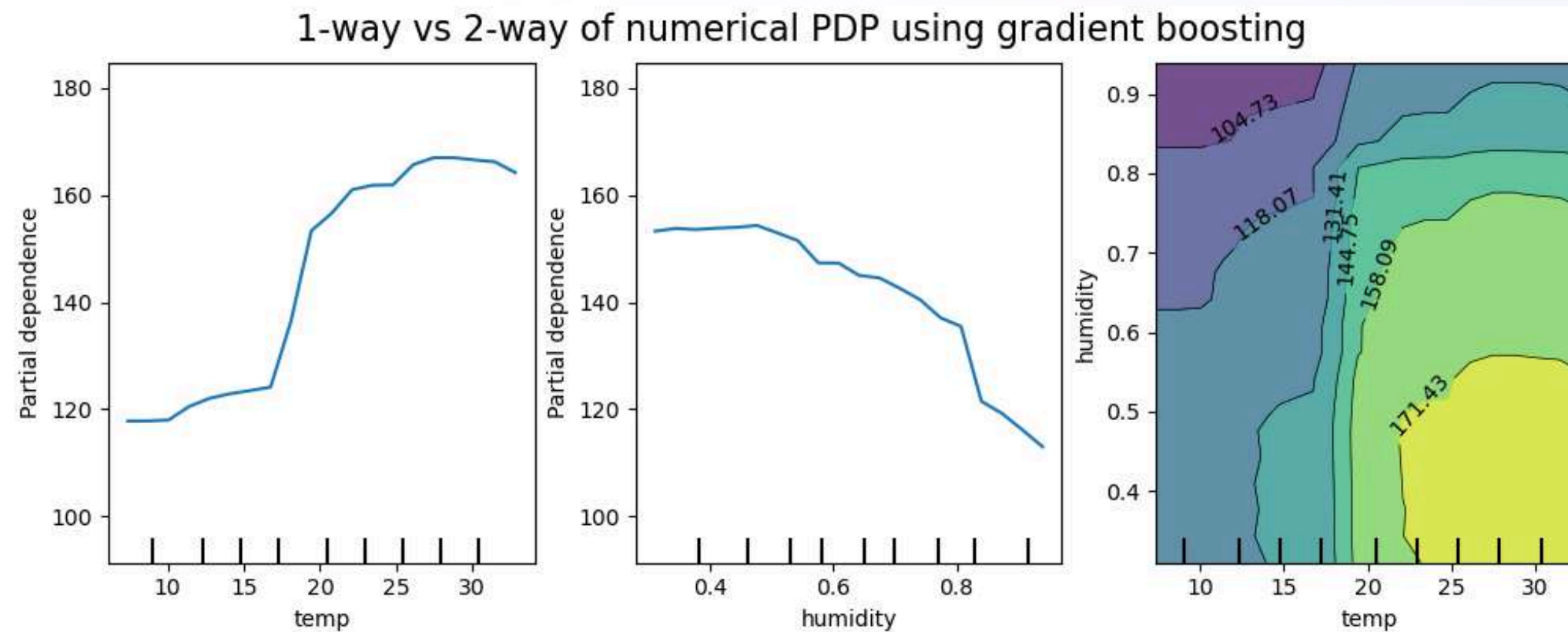
$$\hat{f}_S^{(i)}(x_S) = \hat{f}(x_S, x_C^{(i)})$$

- Each line in an ICE plot represents a single instance, showing how its predicted outcome changes as  $x_S$  varies.
- Reveals whether the relationship between  $x_S$  and the prediction is consistent across instances or if subgroups behave differently.
- If there are too many lines in an ICE plot, it can be difficult to see differences between individual samples and interpret the model. Centering the ICE at the first value on the x-axis, produces centered Individual Conditional Expectation (cICE) plots

**Independence Assumption:** Like PDP, assumes other features remain fixed. this can be misleading if features are correlated.

# Post Hoc Explainability Methods

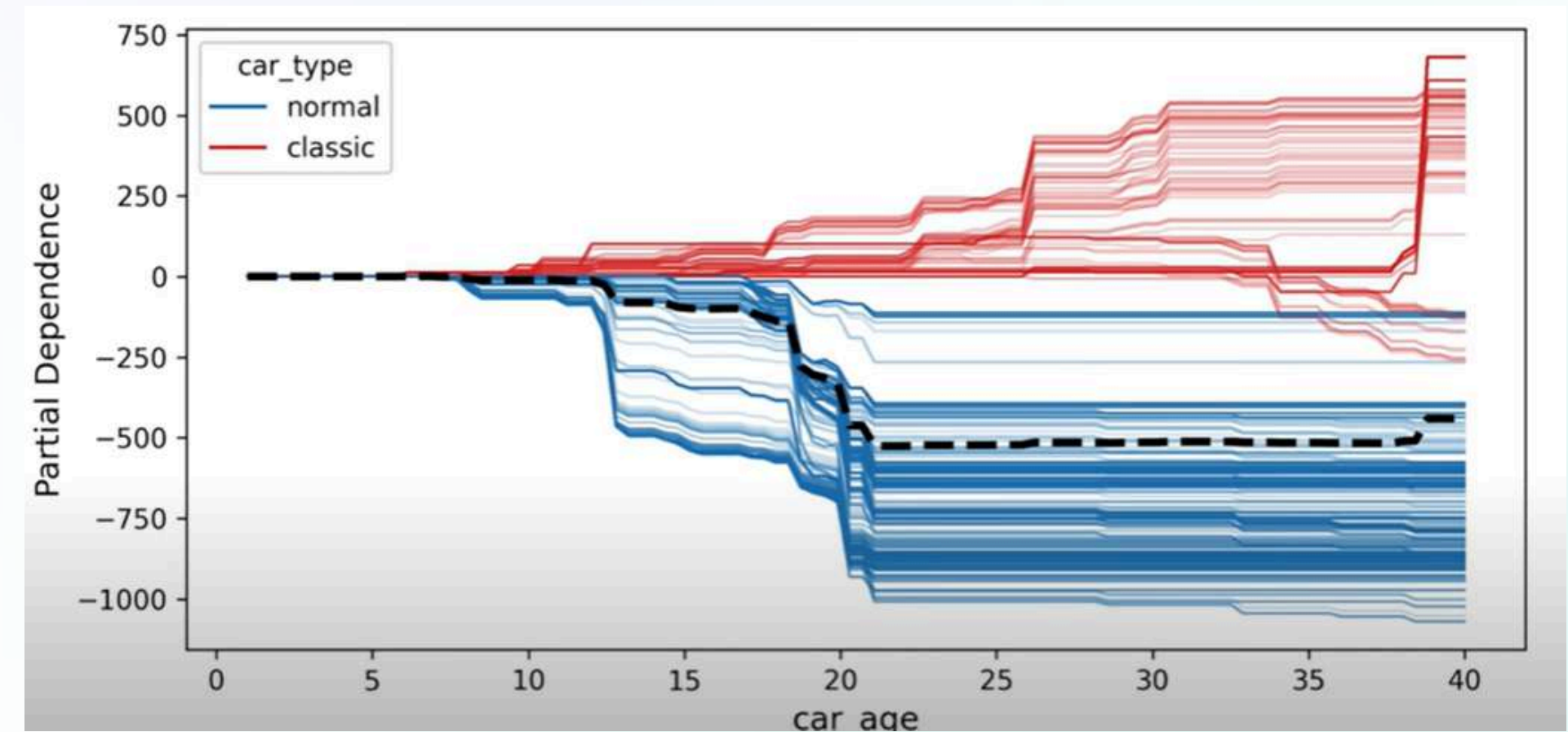
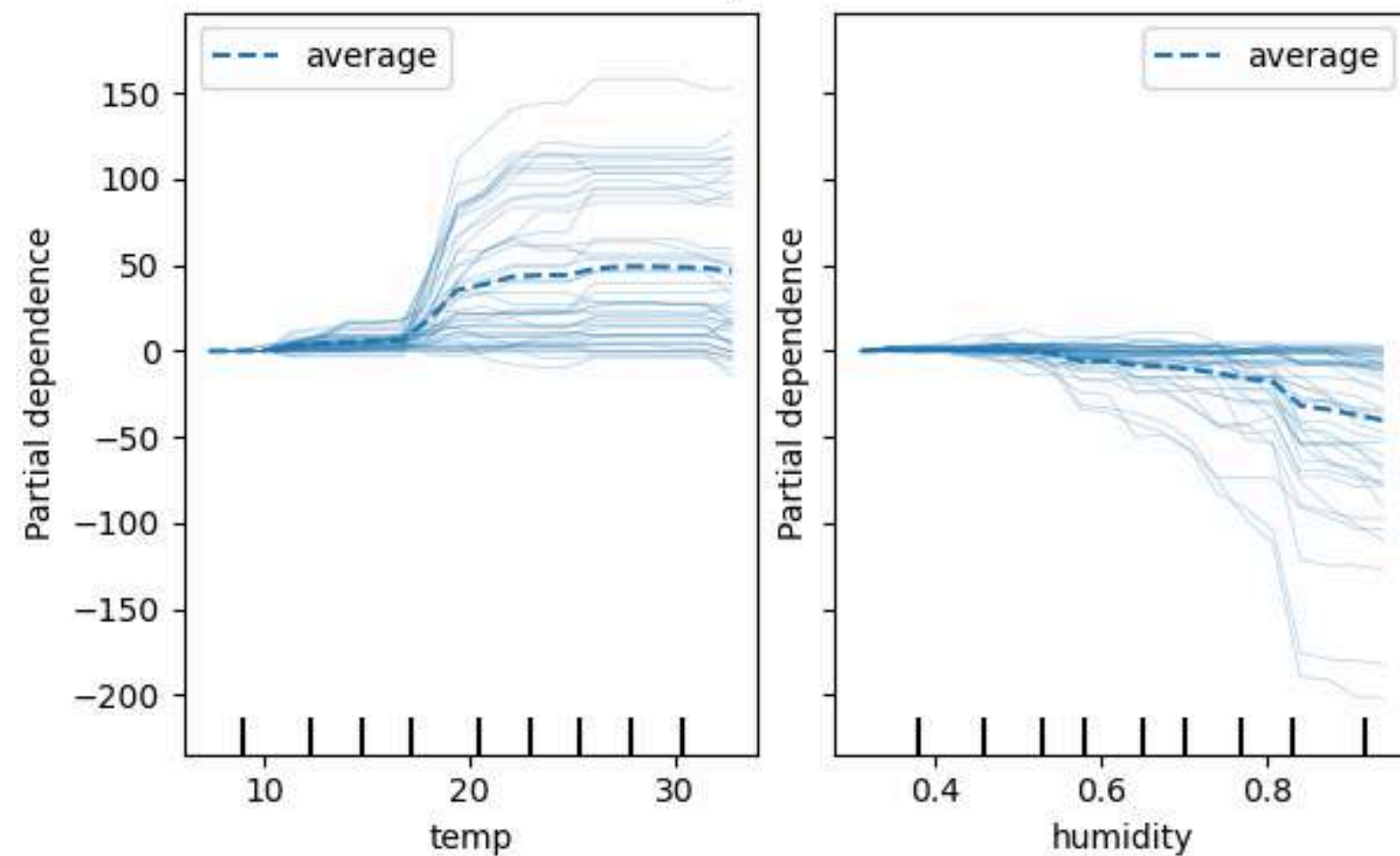
## Feature Importance - PDP & ICE



# Post Hoc Explainability Methods

## Feature Importance - PDP & ICE

ICE and PDP representations



---

# Post Hoc Explainability Methods

## Leave One Feature Out (LOFO) Importance

LOFO (Leave One Feature Out) Importance measures a feature's significance by retraining the model without it and assessing changes in predictive performance.

If removing a feature worsens performance, it is deemed important; if performance remains unchanged, it is not. Negative LOFO importance occurs when removing a feature actually improves model performance. To calculate LOFO importance for all features, the model must be retrained  $p$  times, once for each feature excluded, making it a straightforward algorithm.

**Input:** Trained model  $\hat{f}$ , training data  $(\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$ , test data  $(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}})$ , and error measure  $L$ .

**Procedure:**

1. Measure the original model error:

$$e_{\text{orig}} = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} L(y_{\text{test}}^{(i)}, \hat{f}(\mathbf{x}_{\text{test}}^{(i)}))$$

# Post Hoc Explainability Methods

## Leave One Feature Out (LOFO) Importance

2. For each feature  $j \in \{1, \dots, p\}$ :

- Remove feature  $j$  from the dataset, creating new datasets  $\mathbf{X}_{\text{train},-j}$  and  $\mathbf{X}_{\text{test},-j}$ .
- Train a new model  $\hat{f}_{-j}$  on  $(\mathbf{X}_{\text{train},-j}, \mathbf{y}_{\text{train}})$ .
- Measure the new error on the modified test set:

$$e_{-j} = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} L \left( y_{\text{test}}^{(i)}, \hat{f}_{-j} \left( \mathbf{x}_{\text{test},-j}^{(i)} \right) \right)$$

3 - Calculate LOFO importance for each feature:.

As a quotient:  $LOFO_j = \frac{e_{-j}}{e_{\text{orig}}}$

as a difference:  $LOFO_j = e_{-j} - e_{\text{orig}}$

---

# Post Hoc Explainability Methods

## Leave One Feature Out (LOFO) Importance

- LOFO retrains the model, unlike PFI, which perturbs features in the same model — LOFO captures how the learning algorithm reacts to feature removal.
- Great for feature selection: features with zero or negative LOFO importance can be removed to improve or simplify the model.
- More realistic: LOFO doesn't create artificial data like marginal PFI does.
- More costly: LOFO requires retraining the model for each feature — not ideal for large models or post-hoc audit.
- We can also build confidence intervals thanks to cross validation during the retraining.



# Post Hoc Explainability Methods

## Feature Interaction – H-Statistic for Partial Dependence

The H-statistic quantifies the strength of interaction between two features in a machine learning model. It measures how much the joint partial dependence deviates from the sum of the individual partial dependences.

Let  $x_S = \{x_j, x_k\}$  be the set of two features whose interaction we want to evaluate, and let  $x_C$  be the complement of  $x_S$ . The H-statistic is defined as:

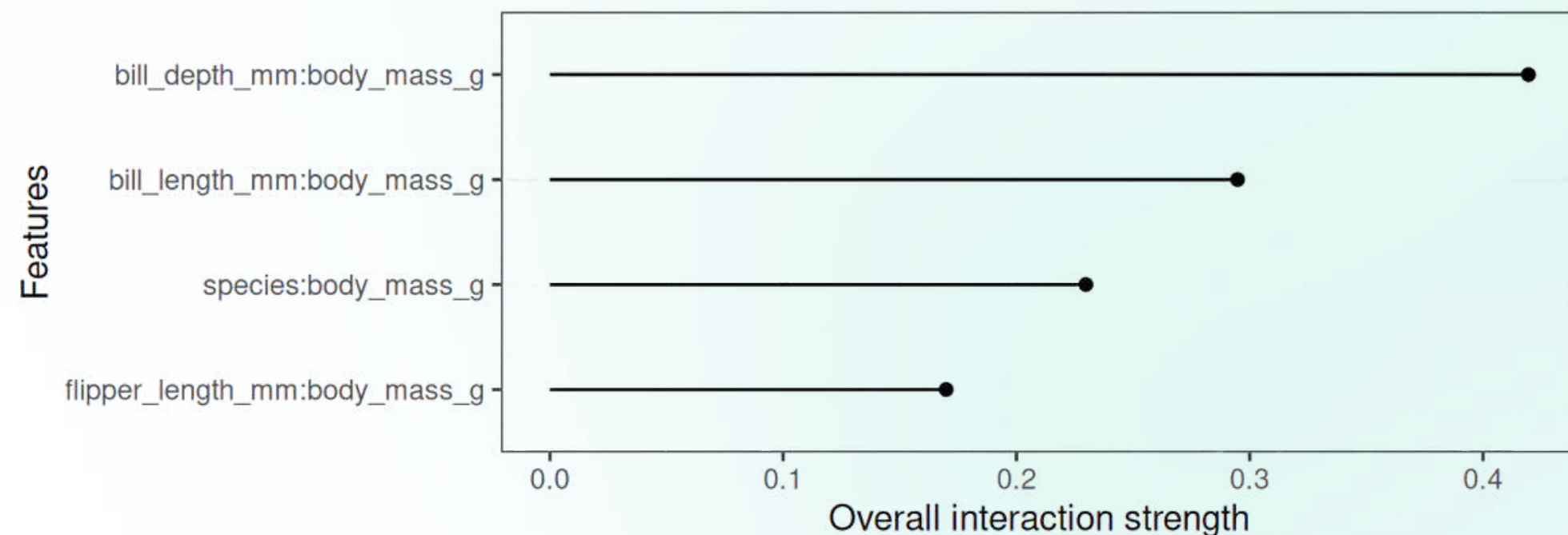
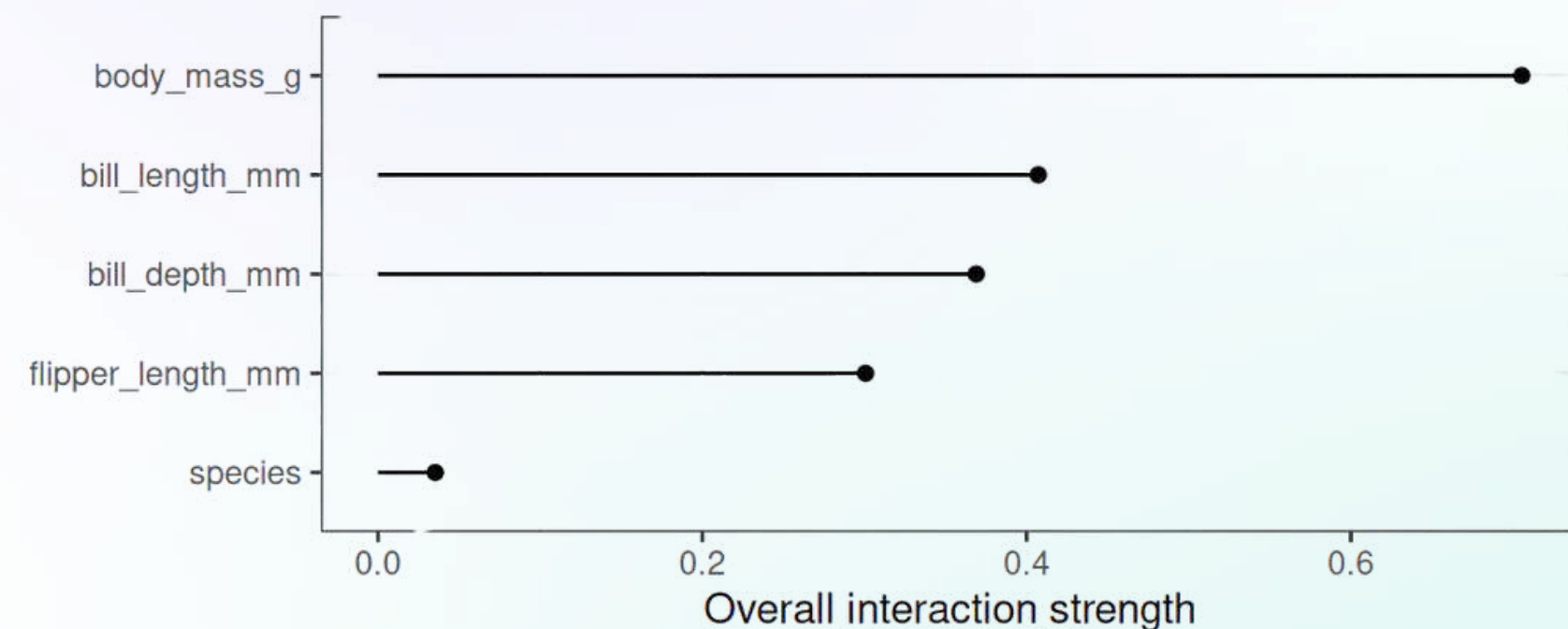
$$H_{jk}^2 = \frac{\sum_{i=1}^n \left( \hat{f}_{j,k}(x_j^{(i)}, x_k^{(i)}) - \hat{f}_j(x_j^{(i)}) - \hat{f}_k(x_k^{(i)}) \right)^2}{\sum_{i=1}^n \hat{f}_{j,k}(x_j^{(i)}, x_k^{(i)})^2}$$

- The H-statistic is costly to evaluate because it involves iterating over all data points and evaluating partial dependence at each point, requiring up to  $2n^2$  calls to the model's predict function in the worst-case scenario.
- To expedite computation, sampling from the  $n$  data points can be used, but this increases the variance of partial dependence estimates, potentially destabilizing the H-statistic. Therefore, ensure adequate sampling to mitigate this effect.

# Post Hoc Explainability Methods

## Feature Interaction – H-Statistic for Partial Dependence

We examine the relationships among features in a random forest model designed to predict the sex of penguins based on their body measurements; Among these features, body mass exhibits the strongest interaction strength.



# Post Hoc Explainability Methods

## Feature Interaction – H-Statistic for Partial Dependence

illustration with linear regression example:  $\hat{f}(x_j, x_k) = \beta_0 + \beta_i x_i + \beta_k x_k + \beta_{jk} x_j x_k$

→ No interaction,

$$\hat{f}(x_j, x_k) = \beta_0 + \beta_i x_i + \beta_k x_k$$

$$H_{jk}^2 = 0$$

→ Strong interaction between features

$$\hat{f}(x_j, x_k) = \beta_0 + \beta_{jk} x_j x_k$$

$$H_{jk}^2 = 1$$

# Post Hoc Explainability Methods

## Feature Interaction – H-Statistic for Partial Dependence

Similarly, if a feature does not interact with any other features, we can represent the prediction function  $\hat{f}(x)$  as a sum of partial dependence functions. In this case, the first term depends solely on feature  $j$ , while the second term accounts for all other features except for  $j$ .

$$H_j^2 = \frac{\sum_{i=1}^n \left( \hat{f}(x^{(i)}) - \hat{f}_j(x_j^{(i)}) - \hat{f}_{-j}(x_{-j}^{(i)}) \right)^2}{\sum_{i=1}^n \hat{f}(x^{(i)})^2}$$

- The H-statistic has a meaningful interpretation: The interaction is defined as the share of variance that is explained by the interaction.
- Since the statistic is dimensionless, it is comparable across features and even across models.
- The statistic detects all kinds of interactions, regardless of their particular form.

---

# Post Hoc Explainability Methods

## Surrogate Models

A global surrogate model is an interpretable model (linear model, decision tree, ...) that is trained to approximate the predictions of a black box model. We can draw conclusions about the black box model by interpreting the surrogate model.

$$g = \arg \min_{g \in \mathcal{G}} \mathbb{E}_{x \sim \mathcal{D}} [L(f(x), g(x))]$$

- Assess the surrogate model's accuracy in replicating the black box model using metrics like R-squared.
- An R-squared close to 1 indicates a strong approximation; consider replacing the complex model with the interpretable one.
- An R-squared near 0 suggests the interpretable model does not effectively explain the black box model.

---

# Post Hoc Explainability Methods

## Local Model-Agnostic Methods

- Local interpretable model-agnostic explanations (LIME)
- Shapley Values
- SHapley Additive exPlanations (SHAP)
- Countrefactual Explanations
- Anchors



---

# Post Hoc Explainability Methods

## Local interpretable model-agnostic explanations (LIME)

A local surrogate model explains individual predictions of any black box model by learning an interpretable model around the neighborhood of the instance to be explained.

LIME builds a dataset of perturbed samples around the instance and weighs them by proximity to the original instance.

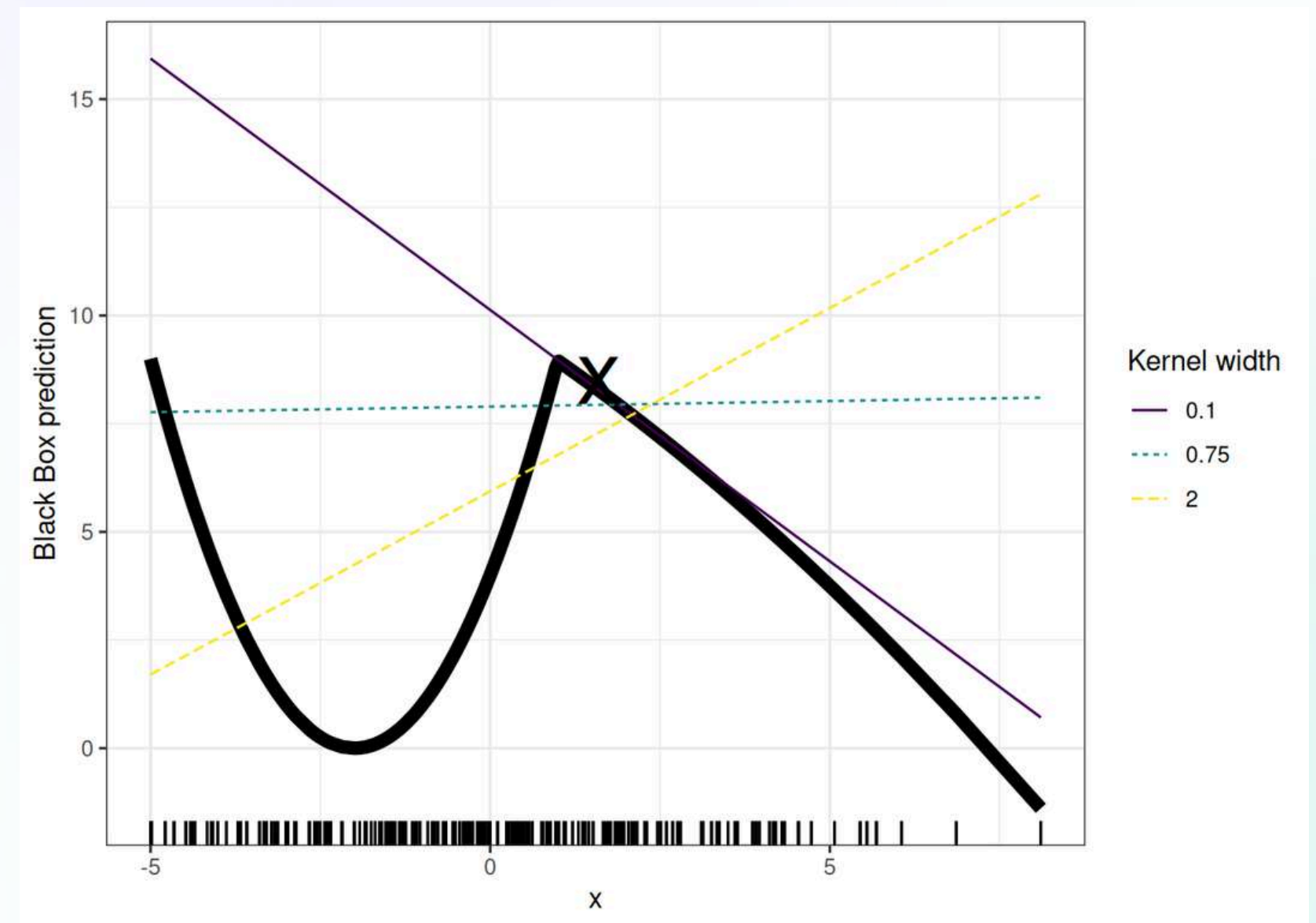
$$g = \arg \min_{g \in \mathcal{G}} \sum_{z \in Z} \pi_x(z) \cdot L(f(z), g(z))$$

- $f$ : the complex model
- $g$ : interpretable model (e.g., linear/lasso)
- $\pi_x(z)$ : proximity measure between  $z$  and the instance  $x$

# Post Hoc Explainability Methods

## LIME for tabular data

- **Perturbation Strategy:** LIME perturbs each feature individually by sampling from a normal distribution centered around the feature's mean and standard deviation.
- **Weighting Samples:** Samples are weighted based on their proximity to the original instance, ensuring that closer samples have a more significant influence on the local surrogate model.
- **Interpretable Model:** A simple model (e.g., linear regression) is trained on the perturbed, weighted samples to approximate the black box model's behavior locally.



# Post Hoc Explainability Methods

## LIME for text data

- Perturbation Strategy: LIME creates variations by randomly removing words from the original text, generating a set of perturbed samples.
- Weighting Samples: Each perturbed text sample is weighted based on its similarity to the original text.
- Interpretable Model: A simple model is trained on these weighted samples to identify which words are most influential in the black box model's prediction

CONTENT	CLASS
PSY is a good guy	0
For Christmas Song visit my channel! ;)	1

For	Christmas	Song	visit	my	channel!	;)	prob	weight
1	0	1	1	0	0	1	0.17	0.57
0	1	1	1	1	0	1	0.17	0.71
1	0	0	1	1	1	1	0.99	0.71
1	0	1	1	1	1	1	0.99	0.86
0	1	1	1	0	0	1	0.17	0.57

# Post Hoc Explainability Methods

LIME for text data

$$\min_w \sum_{i=1}^n \pi_x(x_i) \cdot \left(f(x_i) - w^\top x_i\right)^2$$

case	label_prob	feature	feature_weight
1	0.1701170	is	0.000000
1	0.1701170	good	0.000000
1	0.1701170	a	0.000000
2	0.9939024	channel!	6.180747
2	0.9939024	;)	0.000000
2	0.9939024	visit	0.000000



# Post Hoc Explainability Methods

## LIME for image data

- Perturbation Strategy: LIME segments images into superpixels, creating perturbed samples by toggling these superpixels on or off, using a defined color like gray for inactive ones.
- Weighting Samples: Perturbed images are weighted by their similarity to the original, based on active superpixels.
- Interpretable Model: A simple model is trained on these weighted samples to identify the most influential superpixels in the black box model's prediction.





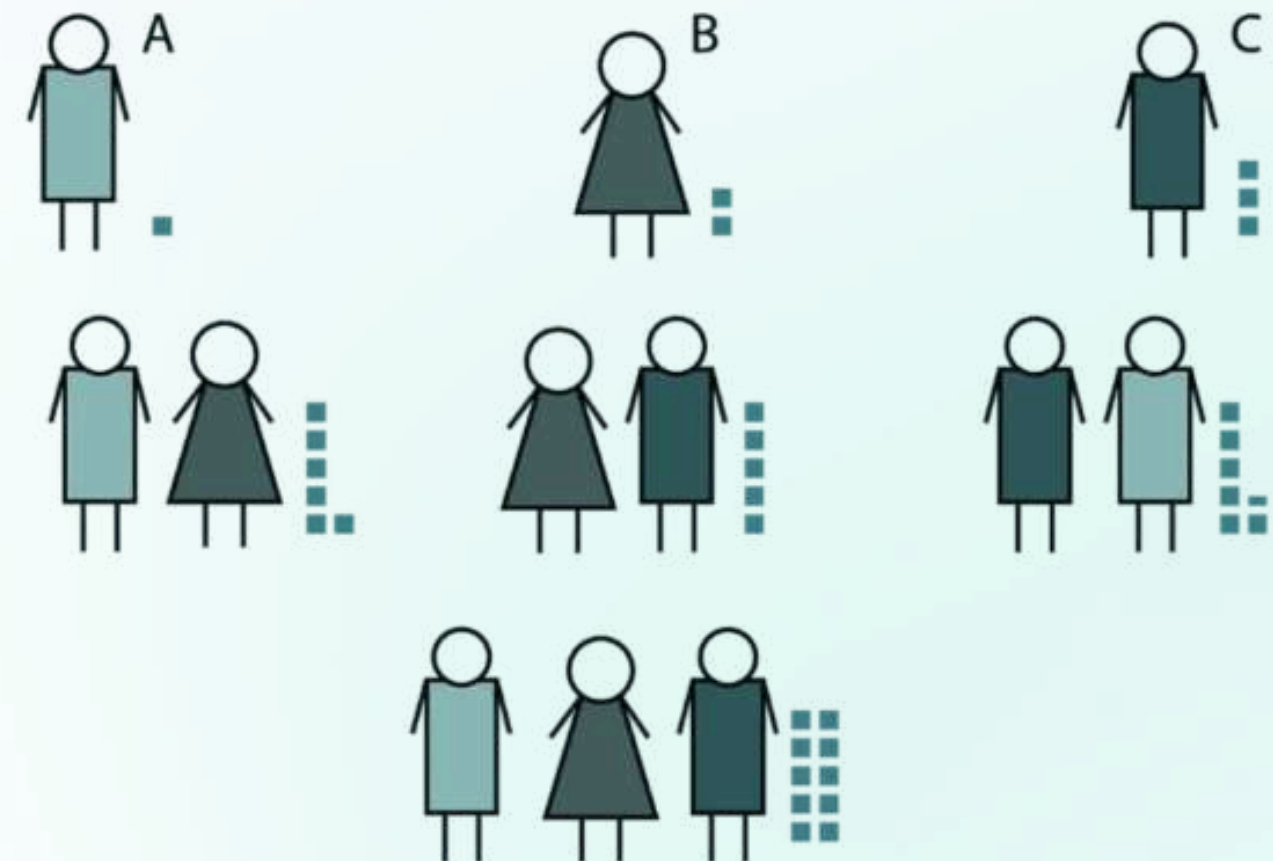
# Post Hoc Explainability Methods

## Shapley Values - General definition in game theory

The Shapley value is a method for fairly distributing total gains or costs among collaborators based on their contributions. It assesses each player's impact by measuring the change in outcomes when they join various combinations of other players, averaging these contributions across all coalitions.

This solution satisfies four key properties: efficiency, symmetry, additivity, and the dummy player property, which are essential for defining fair distribution.

$$\begin{aligned}\varphi_i(v) &= \sum_{S \subseteq N \setminus \{i\}} \frac{|S|! (n - |S| - 1)!}{n!} (v(S \cup \{i\}) - v(S)) \\ &= \frac{1}{n} \sum_{S \subseteq N \setminus \{i\}} \left( \binom{n-1}{|S|} \right)^{-1} (v(S \cup \{i\}) - v(S))\end{aligned}$$




# Post Hoc Explainability Methods

## Shapley Values - General definition in game theory

The value function for this coalitional game is:

$$v(S) = \begin{cases} 1 & \text{if } S \in \{\{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}; \\ 0 & \text{otherwise.} \end{cases}$$

Order $R$	$MC_1$
1, 2, 3	$v(\{1\}) - v(\emptyset) = 0 - 0 = 0$
1, 3, 2	$v(\{1\}) - v(\emptyset) = 0 - 0 = 0$
2, 1, 3	$v(\{1, 2\}) - v(\{2\}) = 0 - 0 = 0$
2, 3, 1	$v(\{1, 2, 3\}) - v(\{2, 3\}) = 1 - 1 = 0$
3, 1, 2	$v(\{1, 3\}) - v(\{3\}) = 1 - 0 = 1$
3, 2, 1	$v(\{1, 3, 2\}) - v(\{3, 2\}) = 1 - 1 = 0$


$$\varphi_1(v) = \left(\frac{1}{6}\right) (1) = \frac{1}{6}.$$

---

# Post Hoc Explainability Methods

## Shapley Values – For features values contribution

Each feature value is treated as a player in a cooperative game, and the model's prediction is the total payout to be distributed.

The Shapley value assigns a fair share of this payout to each feature value by **evaluating its marginal contribution**, that is, how much the feature increases or decreases the prediction compared to the average prediction, across all possible combinations of features.

$$val_x(S) = \int \hat{f}(x_1, \dots, x_p) d\mathbb{P}_{X_C} - \mathbb{E}[\hat{f}(\mathbf{X})]$$

the properties of Shapley Values:

- **Efficiency** The feature contributions must add up to the difference of prediction for  $x$  and the average.
- **Symmetry** The contributions of two feature values  $j$  and  $k$  should be the same if they contribute equally to all possible coalitions.
- **Dummy** A feature  $j$  that does not change the predicted value – regardless of which coalition of feature values it is added to – should have a Shapley value of 0.
- **Additivity** For a game with combined payouts, the respective Shapley values are sums of Shapley values of each game. ex: Random Forest.

# Post Hoc Explainability Methods

## Estimating Shapley values

Evaluating all coalitions of feature values to compute the exact Shapley value is challenging due to the exponential increase in possible coalitions with more features. Štrumbelj and Kononenko (2014) suggested using Monte-Carlo sampling as an approximation method.

$$\hat{\phi}_j = \frac{1}{M} \sum_{m=1}^M \left( \hat{f}(\mathbf{x}_{+j}^{(m)}) - \hat{f}(\mathbf{x}_{-j}^{(m)}) \right)$$

Consider using a smaller sample size  $M$  when estimating Shapley values to reduce computation time, but beware of increased variance in the estimate.

- Output: Shapley value for the value of the  $j$ -th feature
- Required: Number of iterations  $M$ , instance of interest  $\mathbf{x}$ , feature index  $j$ , data matrix  $\mathbf{X}$ , and machine learning model  $f$
- For all  $m = 1, \dots, M$ :
  - Draw random instance  $\mathbf{z}$  from the data matrix  $\mathbf{X}$
  - Choose a random permutation  $\mathbf{o}$  of the feature values
  - Order instance  $\mathbf{x}$ :  $\mathbf{x}_{\mathbf{o}} = (x_{(1)}, \dots, x_{(j)}, \dots, x_{(p)})$
  - Order instance  $\mathbf{z}$ :  $\mathbf{z}_{\mathbf{o}} = (z_{(1)}, \dots, z_{(j)}, \dots, z_{(p)})$
  - Construct two new instances
    - With  $j$ :  $\mathbf{x}_{+j} = (x_{(1)}, \dots, x_{(j-1)}, x_{(j)}, z_{(j+1)}, \dots, z_{(p)})$
    - Without  $j$ :  $\mathbf{x}_{-j} = (x_{(1)}, \dots, x_{(j-1)}, z_{(j)}, z_{(j+1)}, \dots, z_{(p)})$
  - Compute marginal contribution:  $\phi_j^{(m)} = \hat{f}(\mathbf{x}_{+j}) - \hat{f}(\mathbf{x}_{-j})$
- Compute Shapley value as the average:  $\phi_j(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M \phi_j^{(m)}$

# Post Hoc Explainability Methods

## SHapley Additive exPlanations (SHAP)

A player can represent either an individual feature value or a group of feature values, such as superpixels in an image. SHAP innovatively uses Shapley value explanations as an additive feature attribution method, connecting LIME and Shapley values. SHAP defines the explanation as:

$$g(\mathbf{z}') = \phi_0 + \sum_{j=1}^M \phi_j z'_j \quad \mathbf{z}' = (z'_1, \dots, z'_M)^T \in \{0, 1\}^M$$

For  $\mathbf{x}$ , the instance of interest, the coalition vector  $\mathbf{x}'$  is a vector of all 1's, i.e. all feature values are “present”. The formula simplifies to:

$$\hat{f}(\mathbf{x}) = g(\mathbf{x}') = \phi_0 + \sum_{j=1}^M \phi_j x'_j = \mathbb{E}[\hat{f}(X)] + \sum_{j=1}^M \phi_j$$



# Post Hoc Explainability Methods

## SHAP estimation -KernelSHAP

The **KernelSHAP** estimation has five steps:

- Sample coalition vectors  $\mathbf{z}'_k \in \{0, 1\}^M$ ,  $k \in \{1, \dots, K\}$  (1 = feature present in coalition, 0 = feature absent).
- Get prediction for each  $\mathbf{z}'_k$  by first converting  $\mathbf{z}'_k$  to the original feature space and then applying model  $\hat{f}$ :  $\hat{f}(h_{\mathbf{x}}(\mathbf{z}'_k))$ .
- Compute the weight for each coalition  $\mathbf{z}'_k$  with the SHAP kernel.
- Fit weighted linear model.
- Return Shapley values  $\phi_k$ , the coefficients from the linear model.

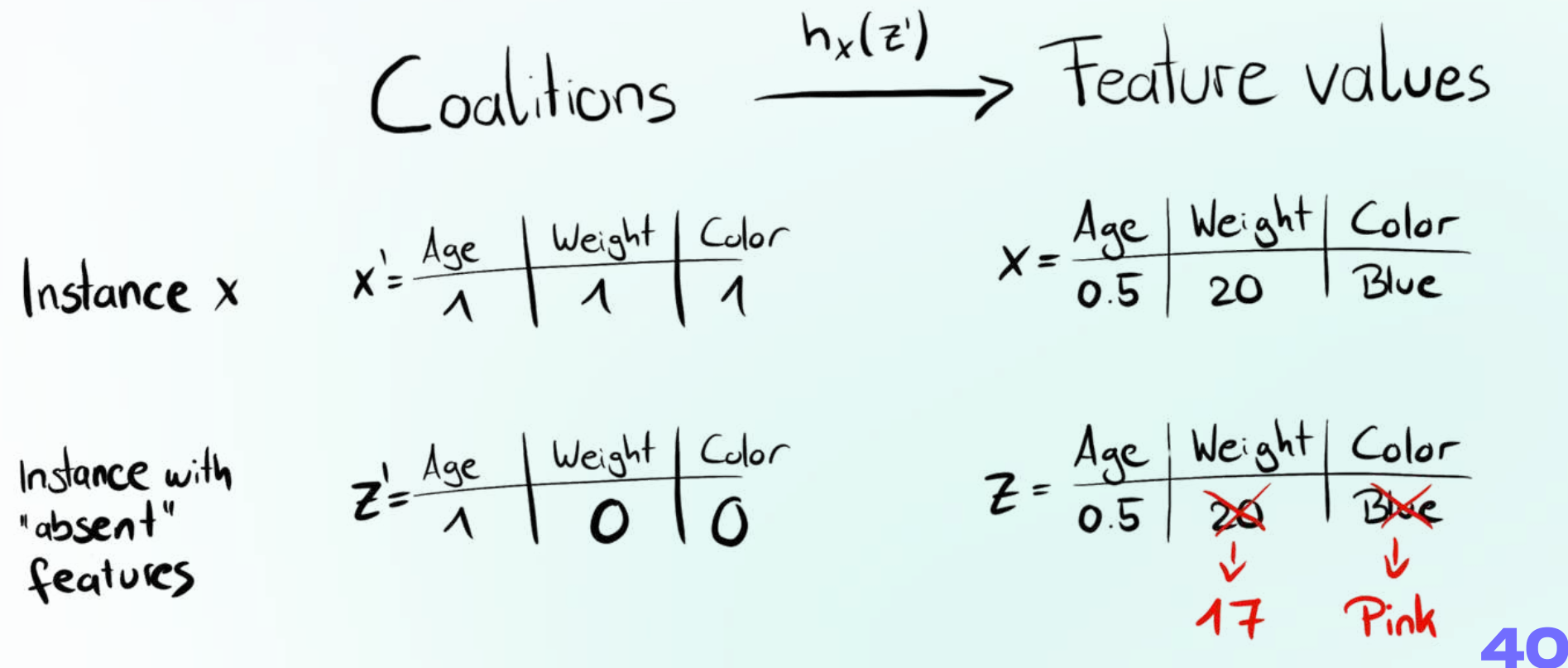
# Post Hoc Explainability Methods

## SHAP estimation -KernelSHAP

The function  $h_x$  maps 1's to the corresponding value from the instance  $x$  that we want to explain. For tabular data, it maps 0's to the values of another instance that we sample from the data. This means that we equate “feature value is absent” with “feature value is replaced by random feature value from data”.

$h_x$  for tabular data treats feature  $X_j$  and  $X_{-j}$  (the other features) as independent and integrates over the marginal distribution:

$$\hat{f}(h_x(\mathbf{z}')) = \mathbb{E}_{X_{-j}}[\hat{f}(x_j, X_{-j})]$$



# Post Hoc Explainability Methods

## SHAP estimation -KernelSHAP

For images, describes a possible mapping function. Assigning the average color of surrounding pixels or similar would also be an option.

SHAP weights the sampled instances according to the weight the coalition would get in the Shapley value estimation

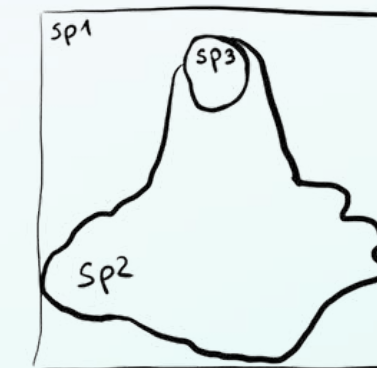
$$\pi_{\mathbf{x}}(\mathbf{z}') = \frac{(M - 1)}{\binom{M}{|\mathbf{z}'|} |\mathbf{z}'| (M - |\mathbf{z}'|)}$$

We train the linear model  $g$  by optimizing the following loss function

$$L(\hat{f}, g, \pi_{\mathbf{x}}) = \sum_{\mathbf{z}' \in \mathcal{Z}} \left[ \hat{f}(h_{\mathbf{x}}(\mathbf{z}')) - g(\mathbf{z}') \right]^2 \pi_{\mathbf{x}}(\mathbf{z}')$$

Coalitions of  $h_{\mathbf{x}}(\mathbf{z}')$   $\rightarrow$  Image

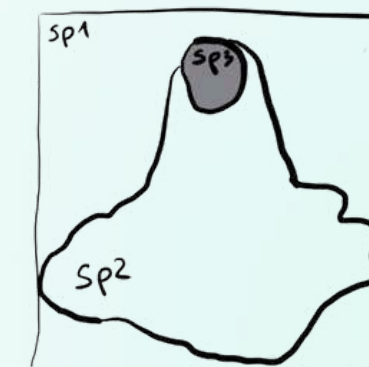
Instance  $x$



sp1	sp2	sp3
1	1	1



Instance  $x$  with absent features



sp1	sp2	sp3
1	1	0



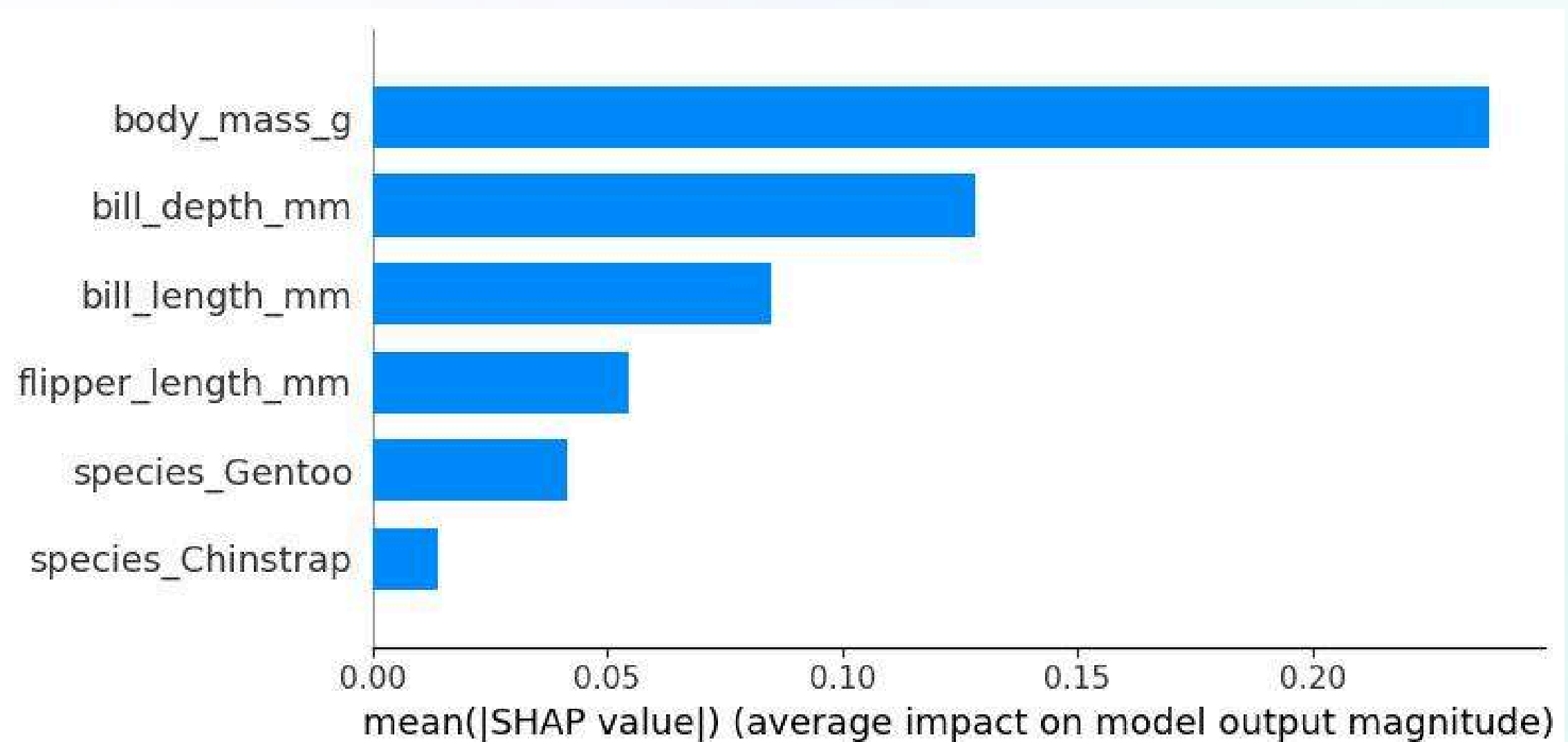
# Post Hoc Explainability Methods

## SHAP aggregation plots - Global version

Shapley values can be aggregated to create comprehensive global explanations. By applying SHAP to each instance, we obtain a matrix of Shapley values. This matrix features one row for each data instance and one column for each feature. By examining the Shapley values within this matrix, we can interpret the entire model effectively.

Since we want the global importance, we average the absolute Shapley values per feature across the data:

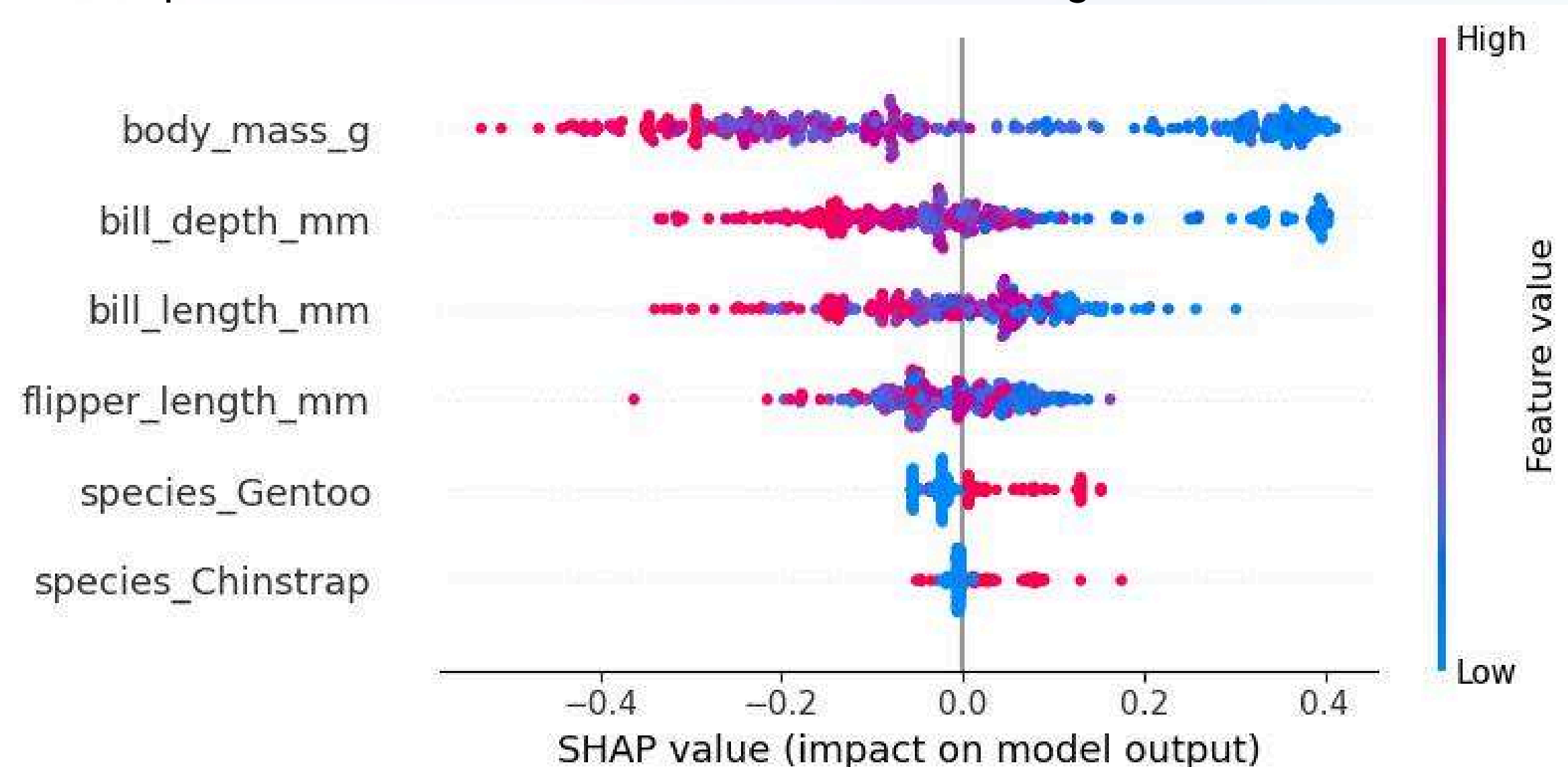
$$I_j = \frac{1}{n} \sum_{i=1}^n |\phi_j^{(i)}|$$



# Post Hoc Explainability Methods

## SHAP Summary Plot

The summary plot combines feature importance with feature effects. Each point on the summary plot is a Shapley value for a feature and an instance. The position on the y-axis is determined by the feature and on the x-axis by the Shapley value. The color represents the value of the feature from low to high.



---

# Post Hoc Explainability Methods

## Global Model-Specific Methods

- GINI Random Forest
- Feature Importance for Gradient Boosted Trees
- Rule Extraction for Tree Ensembles
- Concept Activation Vectors (TCAV)



# Post Hoc Explainability Methods

## GINI Random Forest

Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature.

Gini Impurity

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

$K$  → The number of classes  
 $p_k$  → Proportion of samples of class  $k$  in the node

**Determine Gini Impurity Reduction for Splits:** When a node  $t$  is split into two child nodes  $t_L$  and  $t_R$  using feature  $j$ , the reduction in Gini impurity,  $\Delta I_G(j, t)$ , is calculated as:

$$\Delta I_G(j, t) = I_G(t) - \left( \frac{N_{t_L}}{N_t} I_G(t_L) + \frac{N_{t_R}}{N_t} I_G(t_R) \right)$$

Here,  $N_t$ ,  $N_{t_L}$ , and  $N_{t_R}$  represent the number of samples in the parent node  $t$ , left child node  $t_L$ , and right child node  $t_R$ , respectively.

---

# Post Hoc Explainability Methods

## GINI Random Forest

**Aggregate Gini Reductions Across All Trees:** Sum the Gini impurity reductions for each feature  $j$  across all nodes  $t$  where the feature is used for splitting, and across all trees  $T$  in the forest:

$$\text{Total Reduction}(j) = \sum_T \sum_{t \in T} \Delta I_G(j, t)$$

**Normalize Feature Importances:** Normalize the total reductions to obtain the feature importance scores:

$$\text{Feature Importance}(j) = \frac{\text{Total Reduction}(j)}{\sum_{j'} \text{Total Reduction}(j')}$$

---

# Post Hoc Explainability Methods

## Feature Importance for Gradient Boosted Trees

XGBoost offers three main types of feature importance scores:

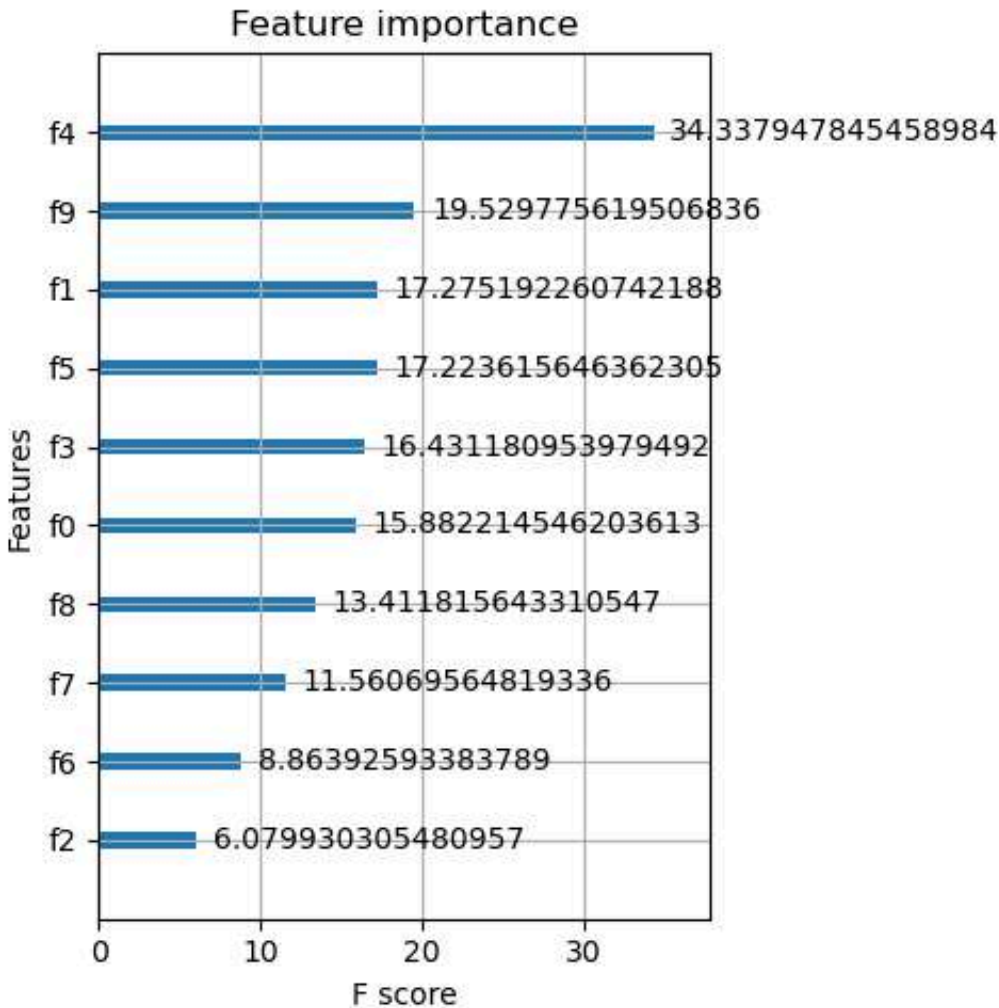
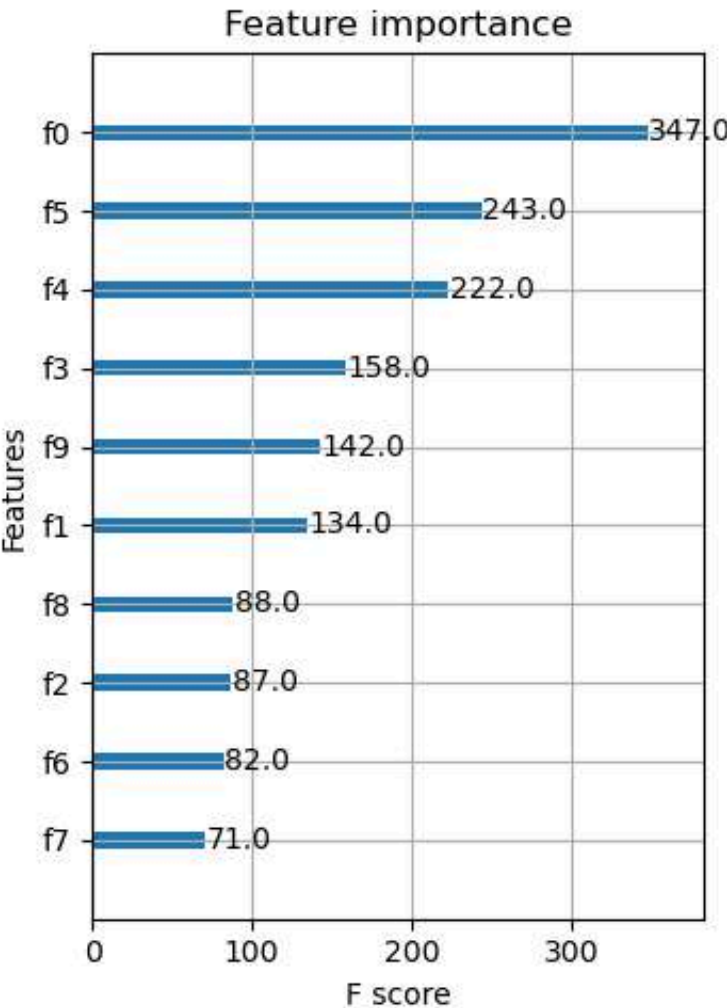
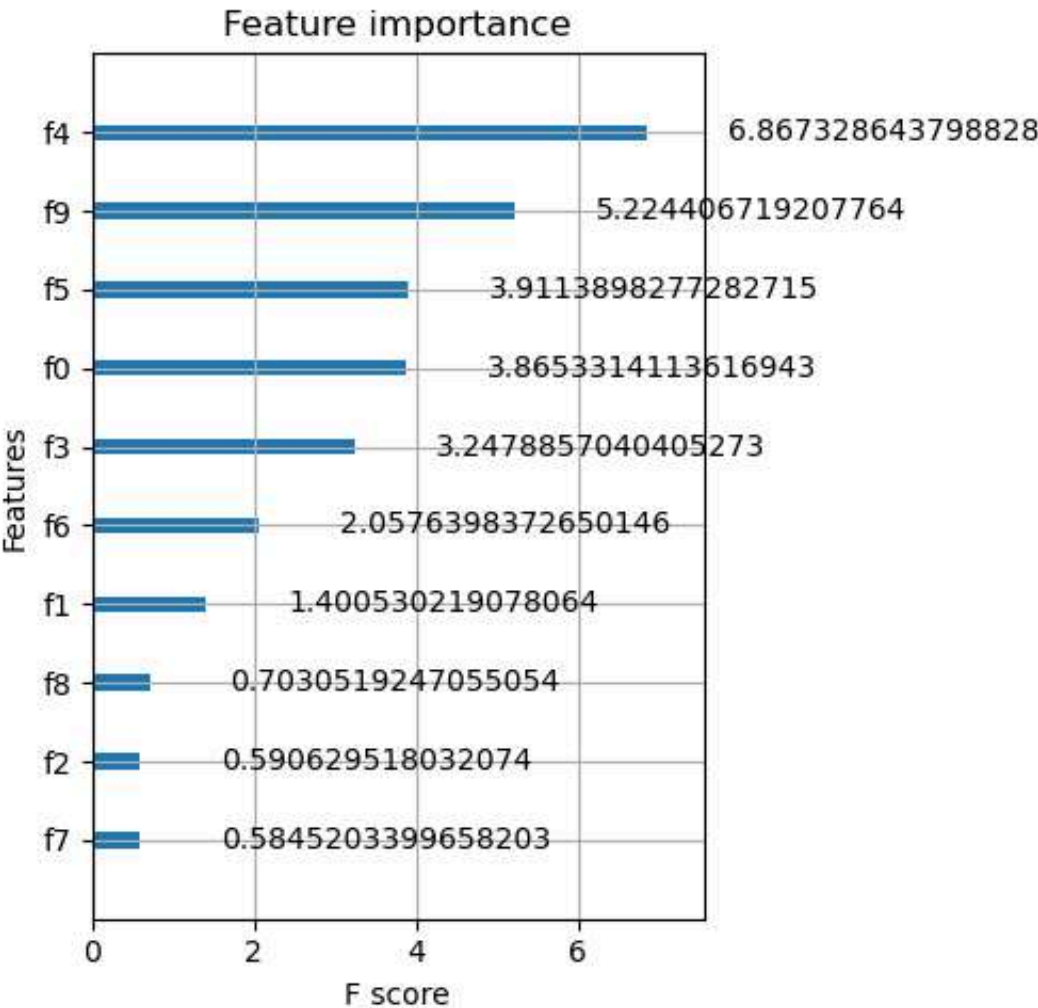
- **Gain-based Importance:** This score measures the average gain of splits that use a particular feature. The gain represents the improvement in accuracy brought by a feature to the branches it is on. Features with higher gain are considered more important. Gain-based importance is useful for assessing a feature's general utility, especially in tree-based models. However, it can be biased towards high-cardinality features.
- **Split-based Importance:** Also known as “weight” or “frequency” importance, this score counts the number of times a feature is used to split the data across all trees in the model. Features used more frequently are deemed more important. Split-based importance is simple and intuitive, making it useful for quick, initial feature filtering or when interpretability is crucial. However, it doesn't account for a feature's actual impact on predictions.
- **Cover-based Importance:** This score measures the average coverage of splits that use the feature. Coverage represents the number of samples affected by the split.

# Post Hoc Explainability Methods

## Feature Importance for Gradient Boosted Trees

When selecting the right feature importance score, keep these guidelines in mind:

- Utilize gain-based and cover-based importance for a general assessment of feature utility, particularly with tree-based models.
- Opt for split-based importance for quick, preliminary feature filtering, especially when interpretability is key.





# Post Hoc Explainability Methods

## Concept Activation Vectors (TCAV)

Testing with Concept Activation Vectors (TCAV) is a method that interprets neural network models by quantifying the influence of high-level, human-friendly concepts on model predictions.

Unlike traditional feature-importance techniques that focus on individual input features, TCAV assesses how user-defined concepts, such as "striped patterns" or "curvature," affect the model's output.

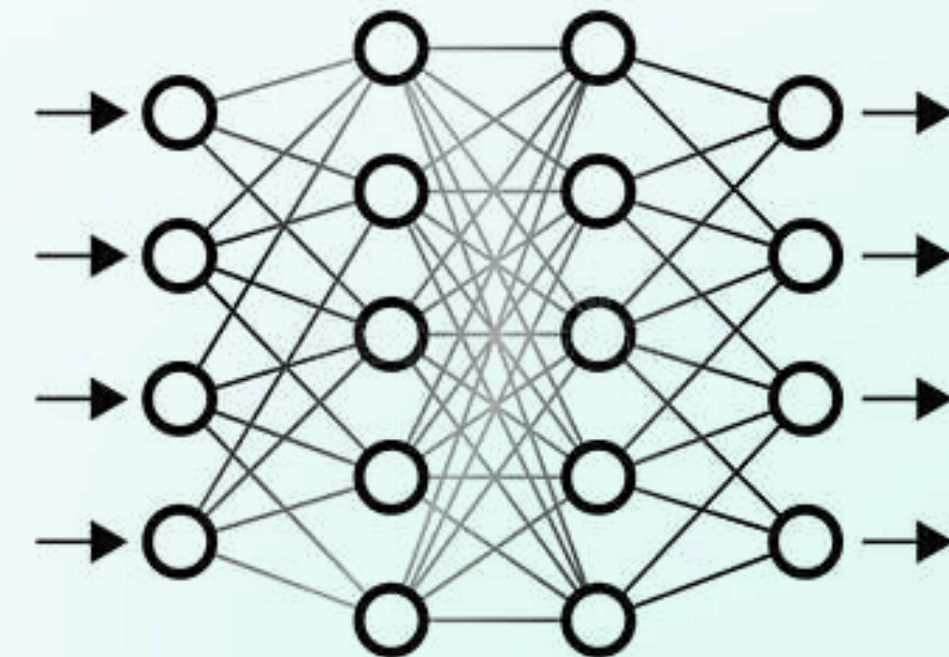
*What is the importance of stripes to identify a tiger?*  
*target image*



+



*concept image*



**Tiger**  
**(0.98)**

# Post Hoc Explainability Methods

## Concept Activation Vectors (TCAV)

Steps to Compute TCAV Scores:

1. Define the Concept and Collect Examples:

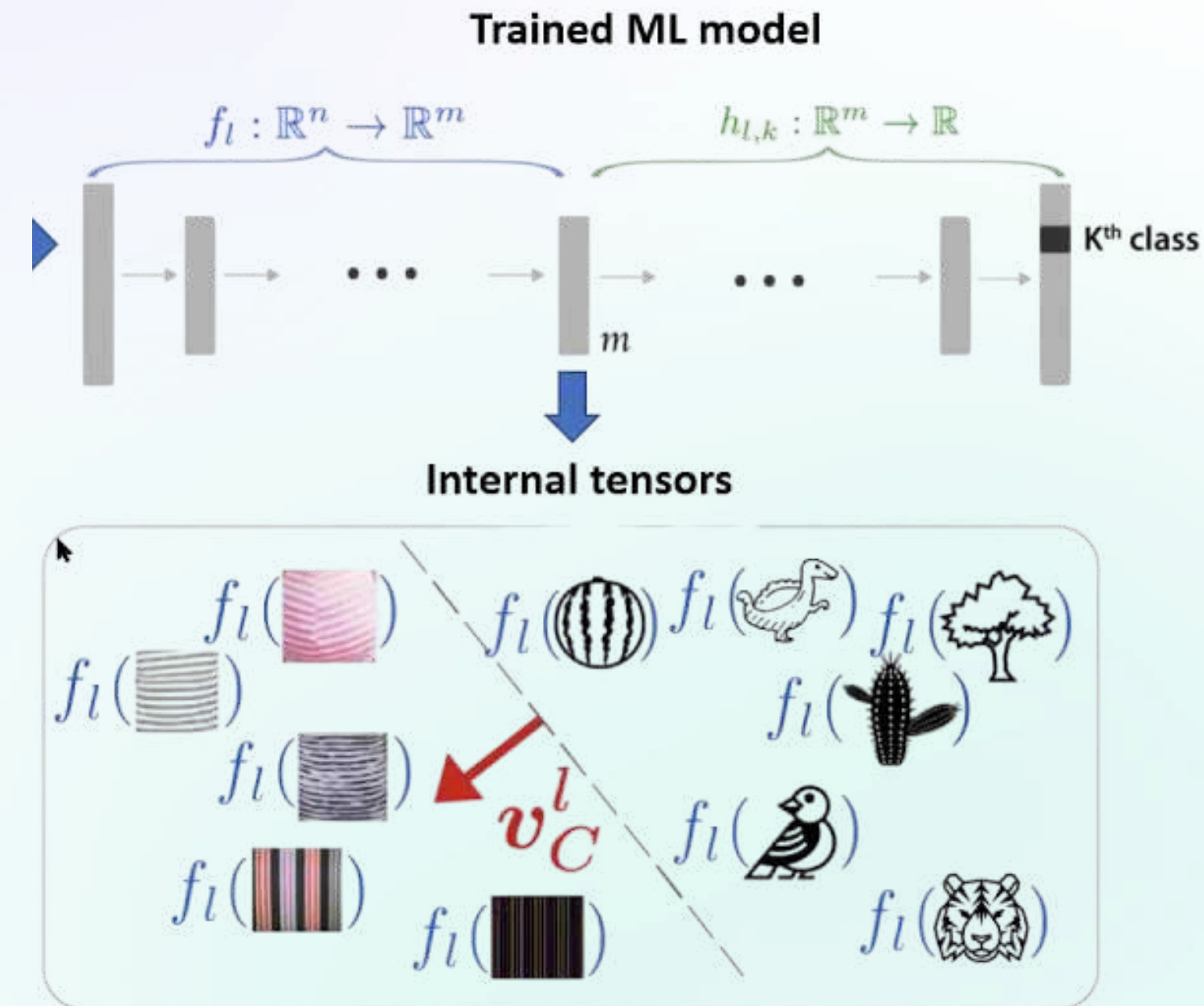
- Identify the concept  $C$  you wish to analyze (e.g., "striped").
- Gather a set of examples that represent this concept, forming the concept dataset  $D_C$ .
- Collect a random set of examples that do not exhibit the concept, forming the random dataset  $D_R$ .

2. Obtain Activations from a Specific Layer:

- Pass both  $D_C$  and  $D_R$  through the neural network up to a chosen layer  $l$  (often an intermediate layer).
- Extract the activation vectors for each example at this layer, resulting in activations  $A_C$  for the concept dataset and  $A_R$  for the random dataset.

3. Train a Linear Classifier to Differentiate the Concept:

- Train a linear classifier (e.g., logistic regression) to distinguish between  $A_C$  and  $A_R$ .
- The normal vector  $\mathbf{v}_l^C$  of the decision boundary learned by this classifier serves as the Concept Activation Vector (CAV) for concept  $C$  at layer  $l$ .



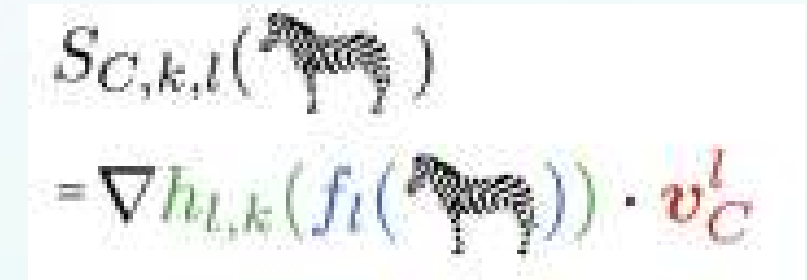


# Post Hoc Explainability Methods

## Concept Activation Vectors (TCAV)

4 - When provided with an image input  $\mathbf{x}$ , we can assess its "conceptual sensitivity" by calculating the directional derivative of the prediction along the direction of the unit CAV.

$$S_{C,k,l}(\mathbf{x}) = \nabla h_{l,k}(\hat{f}_l(\mathbf{x})) \cdot \mathbf{v}_l^C$$


$$S_{C,k,l}(\text{zebra}) = \nabla h_{l,k}(f_l(\text{zebra})) \cdot \mathbf{v}_C^l$$

$h_{l,k}$  maps the activation vector to the logit output of class  $k$  at layer  $l$ .

$S_{C,k,l}(\mathbf{x}) > 0$  can be interpreted as concept  $C$  encouraging the model to classify  $\mathbf{x}$  into class  $k$ .

5 - Calculate the TCAV Score:

$$TCAV_{Q,C,k,l} = \frac{|\{\mathbf{x} \in \mathbf{X}_k : S_{C,k,l}(\mathbf{x}) > 0\}|}{|\mathbf{X}_k|}$$

The TCAV score for the concept "striped" with the class "zebra" is determined by the ratio of zebra images with positive conceptual sensitivities to the total number of zebra images.

---

# Post Hoc Explainability Methods

## Local Model-Specific Methods

- Saliency Maps (Gradient-Based Explanations)
- Grad-CAM (Gradient-Weighted Class Activation Mapping)
- Layer-wise Relevance Propagation (LRP)
- DeepLIFT

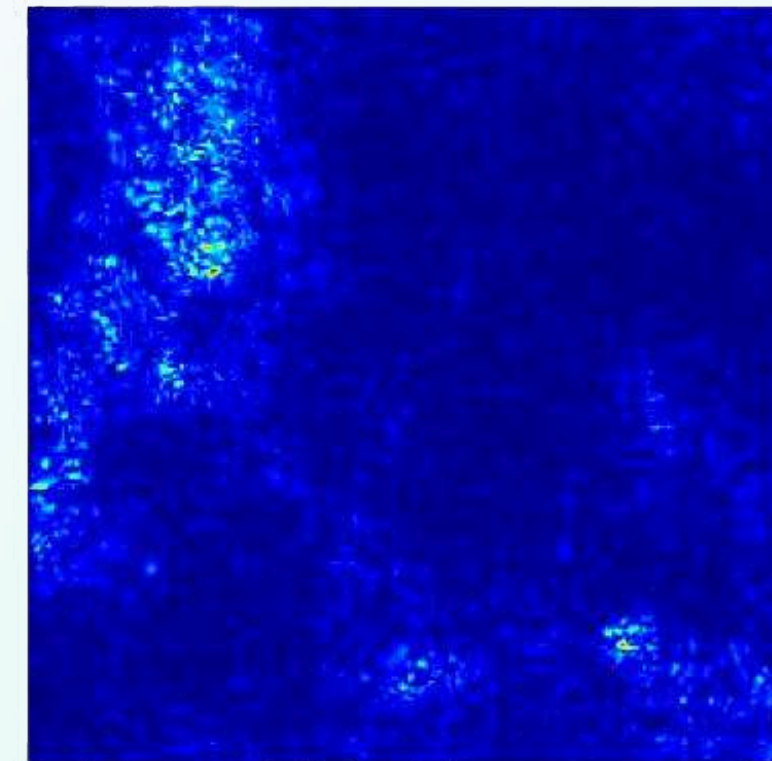
# Post Hoc Explainability Methods

## Saliency Maps - Vanilla Gradient

We calculate the gradient of the loss function for the class we are interested in with respect to the input pixels. This gives us a map of the size of the input features with negative to positive values.

- Perform a forward pass of the image of interest.
- Compute the gradient of the class score of interest with respect to the input pixels.
- Visualize the gradients. You can either show the absolute values or highlight negative and positive contributions separately.

$$E_{grad}(\mathbf{x}_0) = \frac{\delta S_c}{\delta \mathbf{X}} \big|_{\mathbf{x}=\mathbf{x}_0}$$



---

# Post Hoc Explainability Methods

## Grad-CAM

Grad-CAM is a technique that visually explains decisions made by Convolutional Neural Networks (CNNs) by highlighting influential regions in input images for specific class predictions. It enhances interpretability by identifying parts of an image that contribute to the model's decision, providing insights into its reasoning process.

To understand how the CNN makes decisions, Grad-CAM analyzes which regions are activated in the feature maps of the last convolutional layers.

Grad-CAM has to decide how important each of the  $k$  feature maps was to our class  $c$ , that we are interested in. We have to weight each pixel of each feature map with the gradient before we average over the feature maps. This gives us a heatmap which highlights regions that positively or negatively affect the class of interest. This heatmap is sent through the ReLU function, which is a fancy way of saying that we set all negative values to zero.

$$L_{\text{Grad-CAM}}^c \in \mathbb{R}^{U \times V} = \text{ReLU} \left( \sum_k \alpha_k^c A^k \right)$$

# Post Hoc Explainability Methods

## Grad-CAM

$$L_{\text{Grad-CAM}}^c \in \mathbb{R}^{U \times V} = \text{ReLU} \left( \sum_k \alpha_k^c A^k \right)$$

Here,  $Z$  is the number of pixels in the feature map,

$A^k$  denotes the  $k$ -th feature map of the last convolutional layer.

Here,  $U$  is the width,  $V$  the height of the explanation, and  $c$  the class of interest.

1. Forward-propagate the input image through the convolutional neural network.
2. Obtain the raw score for the class of interest, meaning the activation of the neuron before the softmax layer.
3. Set all other class activations to zero.
4. Back-propagate the gradient of the class of interest to the last convolutional layer before the fully connected layers:  $\frac{\delta y^c}{\delta A^k}$ .
5. Weight each feature map “pixel” by the gradient for the class. Indices  $u$  and  $v$  refer to the width and height dimensions:

$$\alpha_k^c = \overbrace{\frac{1}{Z} \sum_u \sum_v}^{\text{global average pooling}} \underbrace{\frac{\delta y^c}{\delta A_{uv}^k}}_{\text{gradients via backprop}}$$

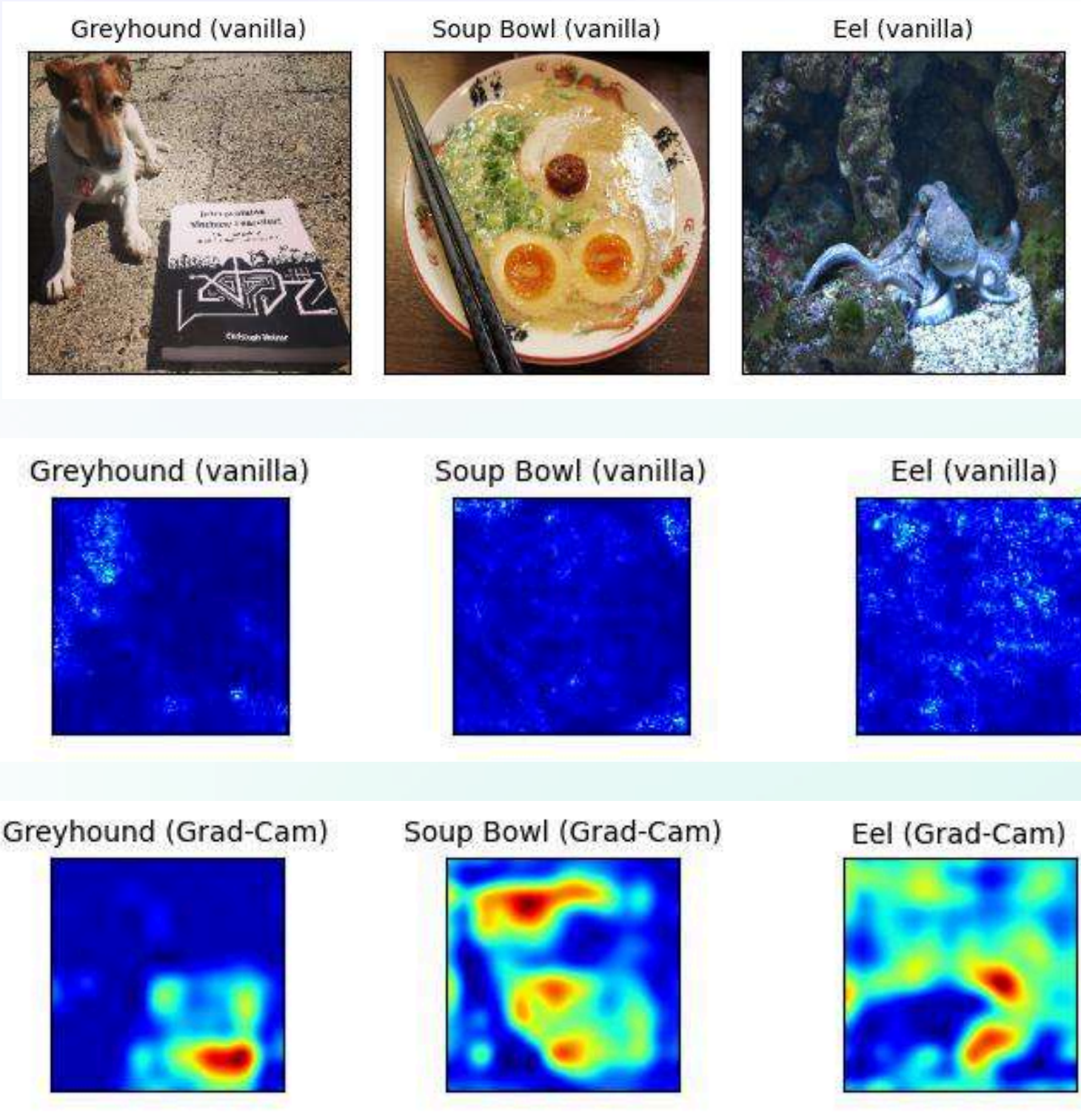
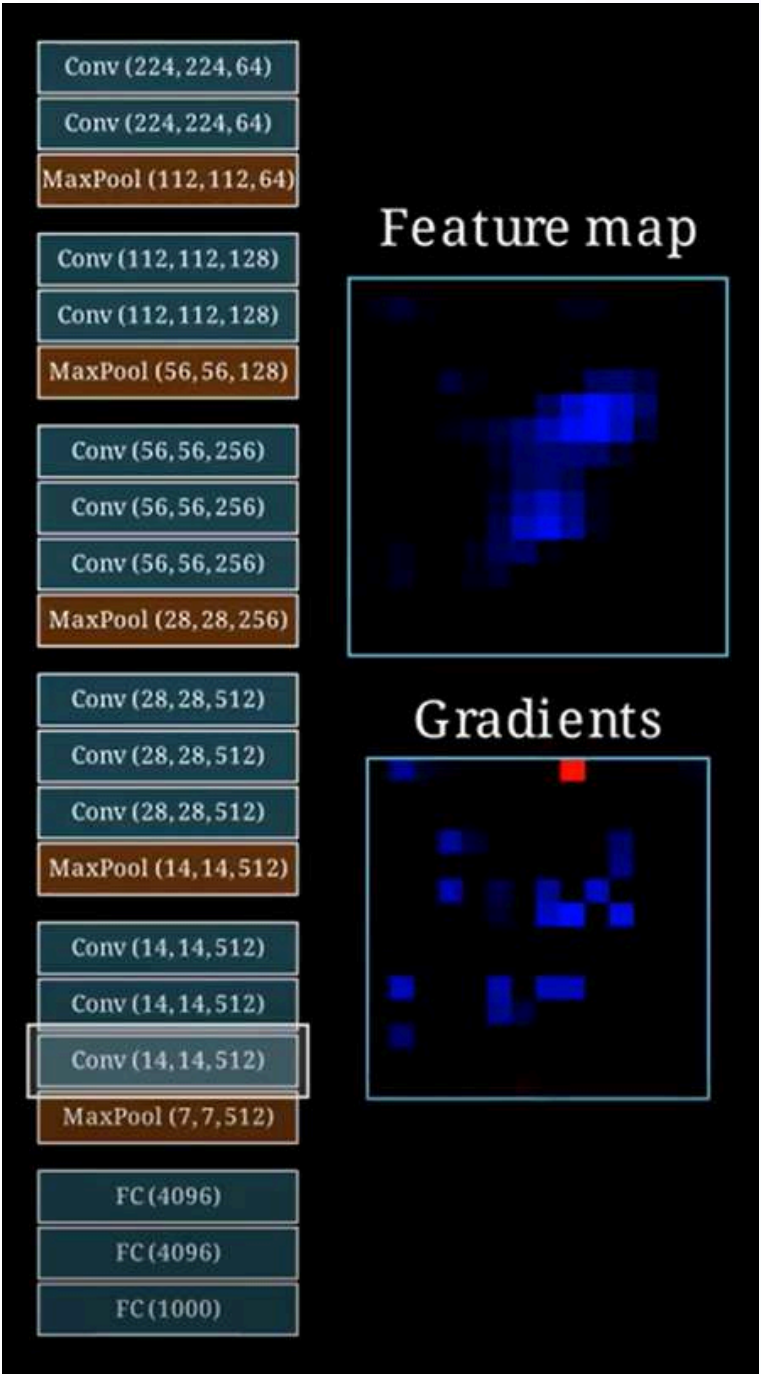
This means that the gradients are globally pooled.

6. Calculate an average of the feature maps, weighted per pixel by the gradient.
7. Apply ReLU to the averaged feature map.
8. For visualization: Scale values to the interval  $[0, 1]$ . Upscale the image and overlay it over the original image.
9. Additional step for Guided Grad-CAM: Multiply heatmap with guided backpropagation.



# Post Hoc Explainability Methods

## Examples

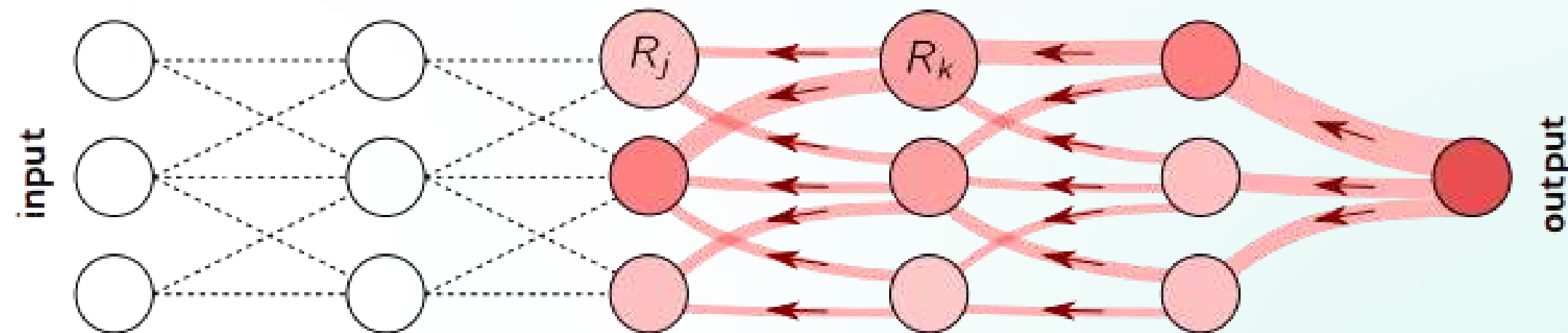


# Post Hoc Explainability Methods

## Layer-wise Relevance Propagation (LRP)

Layer-wise Relevance Propagation (LRP) is a technique that explains individual predictions of a neural network by propagating the prediction score backward through the layers of the network, distributing a relevance score to each input feature.

To identify and quantify which input features (e.g., pixels, words, variables) contributed most to a specific prediction, thereby improving the interpretability of deep models.



### Key Concept:

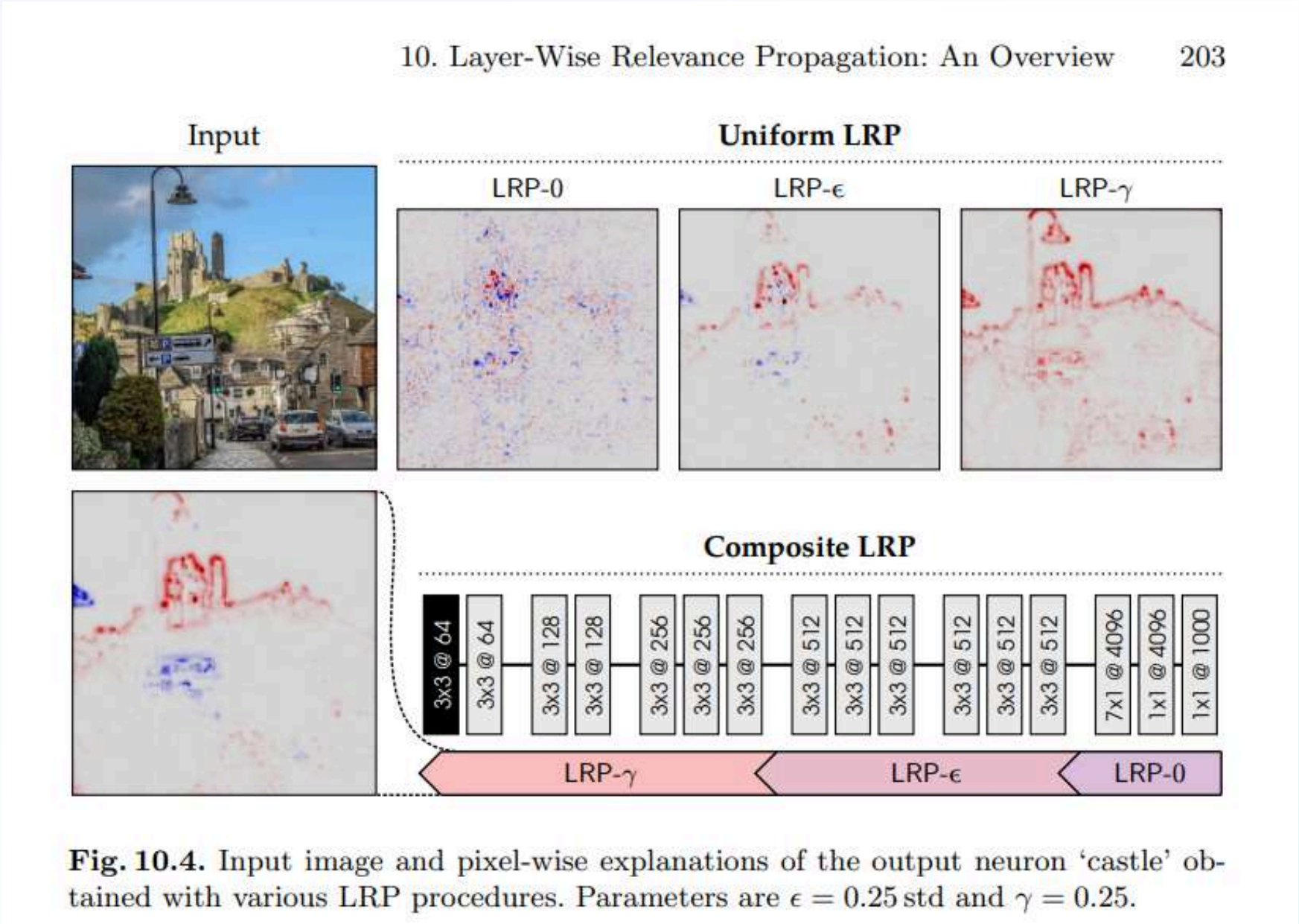
- Unlike gradient-based methods that measure sensitivity, LRP attributes the actual prediction value back to the input features.
- It preserves a conservation principle: the sum of the relevance scores at each layer equals the output score of the model.



# Post Hoc Explainability Methods

## Layer-wise Relevance Propagation (LRP)

Name	Formula	Usage
LRP-0 [7]	$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k$	upper layers
LRP- $\epsilon$ [7]	$R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \sum_{0,j} a_j w_{jk}} R_k$	middle layers
LRP- $\gamma$	$R_j = \sum_k \frac{a_j (w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j (w_{jk} + \gamma w_{jk}^+)} R_k$	lower layers
LRP- $\alpha\beta$ [7]	$R_j = \sum_k \left( \alpha \frac{(a_j w_{jk})^+}{\sum_{0,j} (a_j w_{jk})^+} - \beta \frac{(a_j w_{jk})^-}{\sum_{0,j} (a_j w_{jk})^-} \right) R_k$	lower layers
flat [30]	$R_j = \sum_k \frac{1}{\sum_j 1} R_k$	lower layers
$w^2$ -rule [36]	$R_i = \sum_j \frac{w_{ij}^2}{\sum_i w_{ij}^2} R_j$	first layer ( $\mathbb{R}^d$ )
$z^B$ -rule [36]	$R_i = \sum_j \frac{x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-}{\sum_i x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-} R_j$	first layer (pixels)



**Fig. 10.4.** Input image and pixel-wise explanations of the output neuron ‘castle’ obtained with various LRP procedures. Parameters are  $\epsilon = 0.25$  std and  $\gamma = 0.25$ .