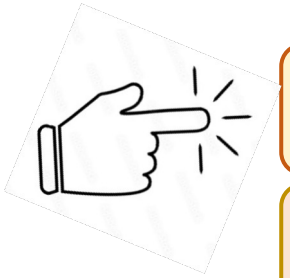
AI VIET NAM
@aivietnam.edu.vn

XAI and LIME
(Explainable AI)

Vinh Dinh Nguyen - PhD in Computer Science
Phuc-Thinh Nguyen - STA
Huy-Duc Tran - STA

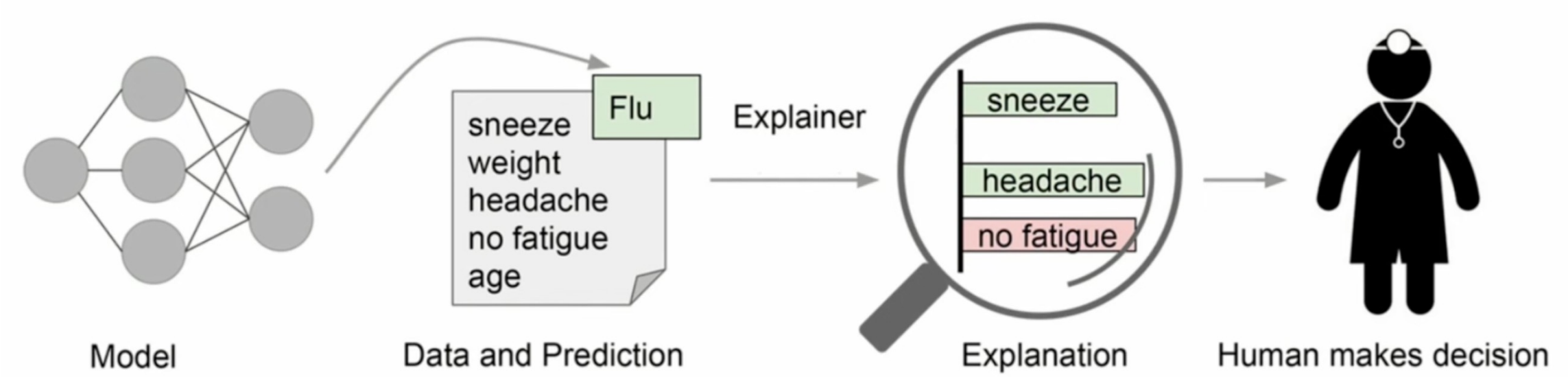
Data & Code

Outline



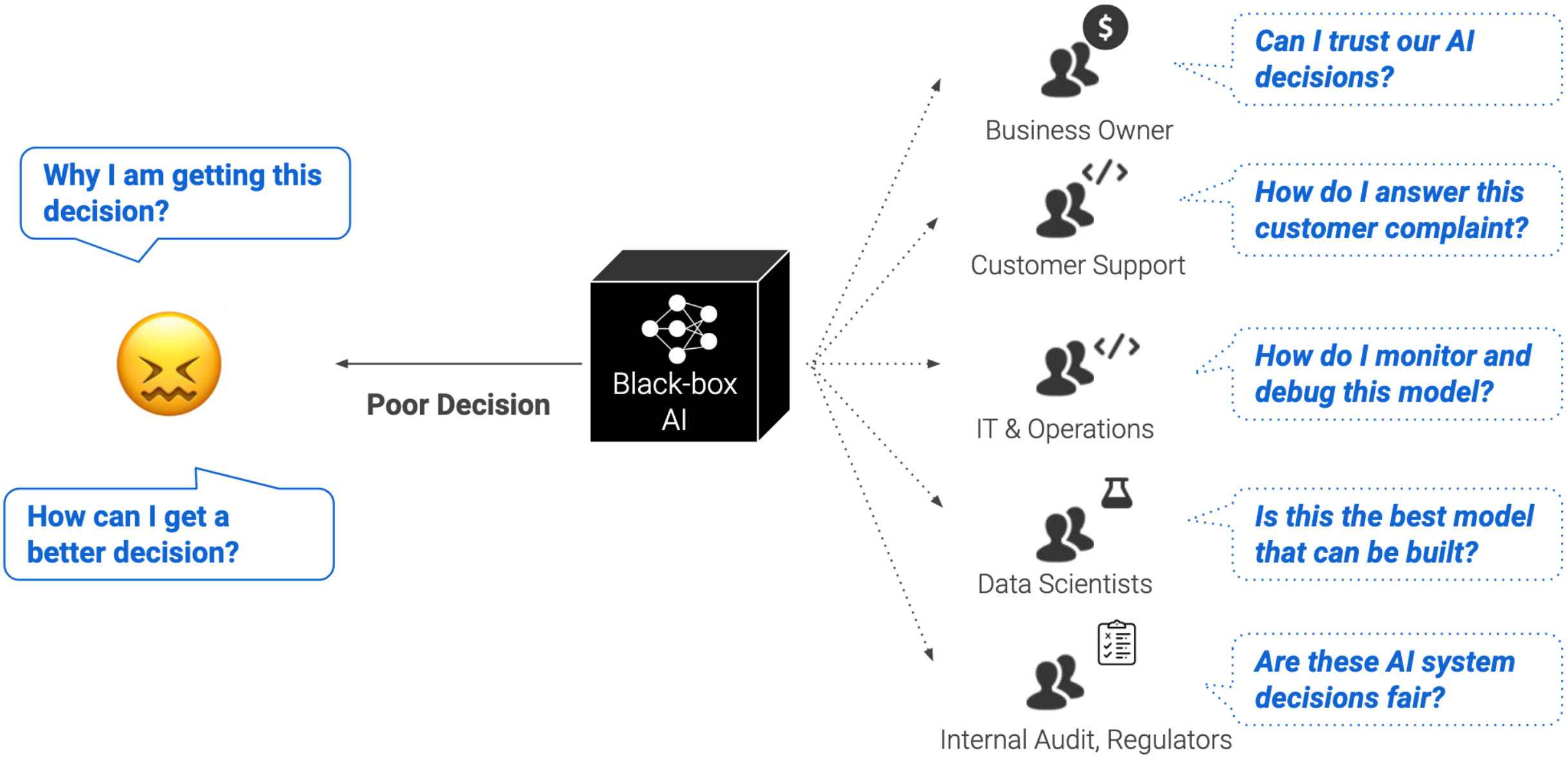
- Why is Interpretability Need?
- Evolution of XAI
- LIME for AI Explainable
- LIME: Example

Overview of Explainable AI



The model needs explanations for humans to trust its decision

Overview of Explainable AI



Overview of Explainable AI

Transparency

Model transparency: How clearly can the inner workings of a model (e.g., weights, features, and decision rules) be observed and understood?

Process transparency: Can a human understand how data is collected, processed, and used to make predictions or decisions?

Decision transparency: Can a user understand why the AI made a specific decision or recommendation?

"Can I see how it works?"

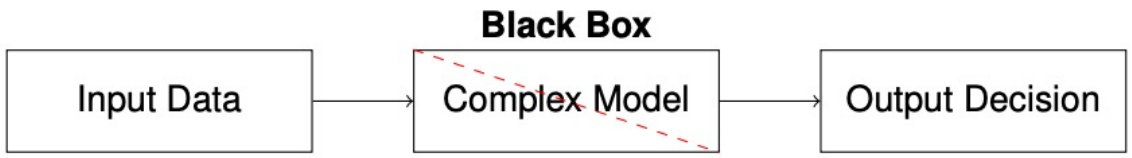
Interpretability

If transparency is about **seeing inside the model**, then interpretability is about **making sense of what you see**.

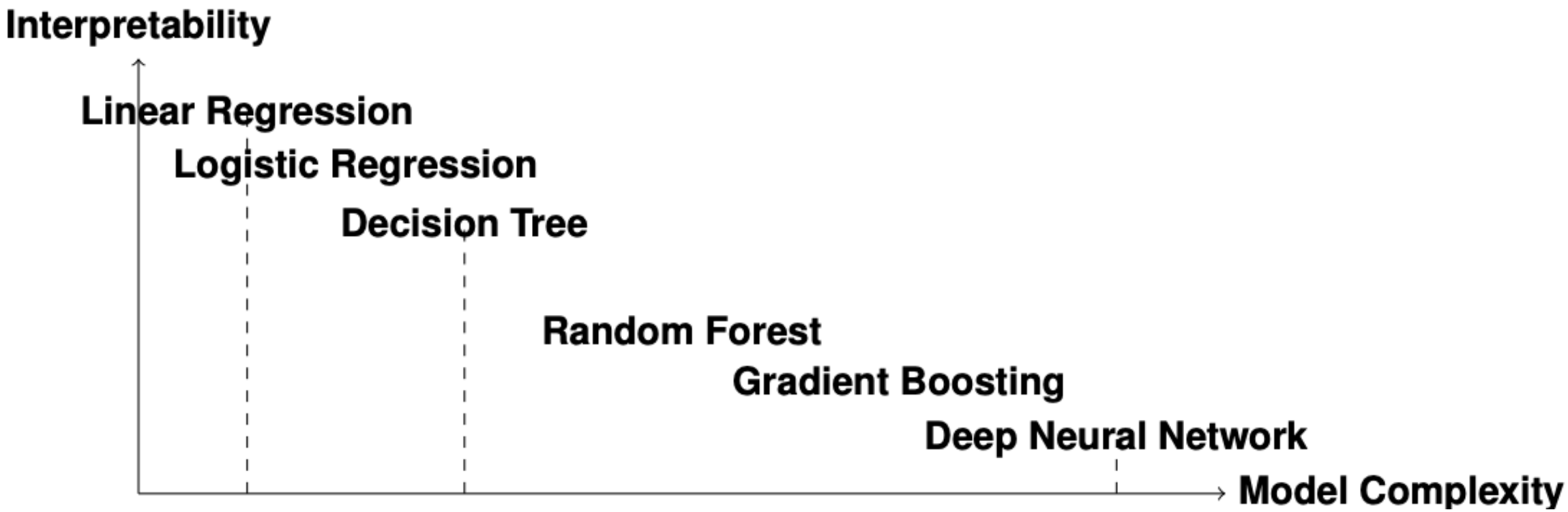
"Can I understand why it did that?"

Why is Interpretability Needed?

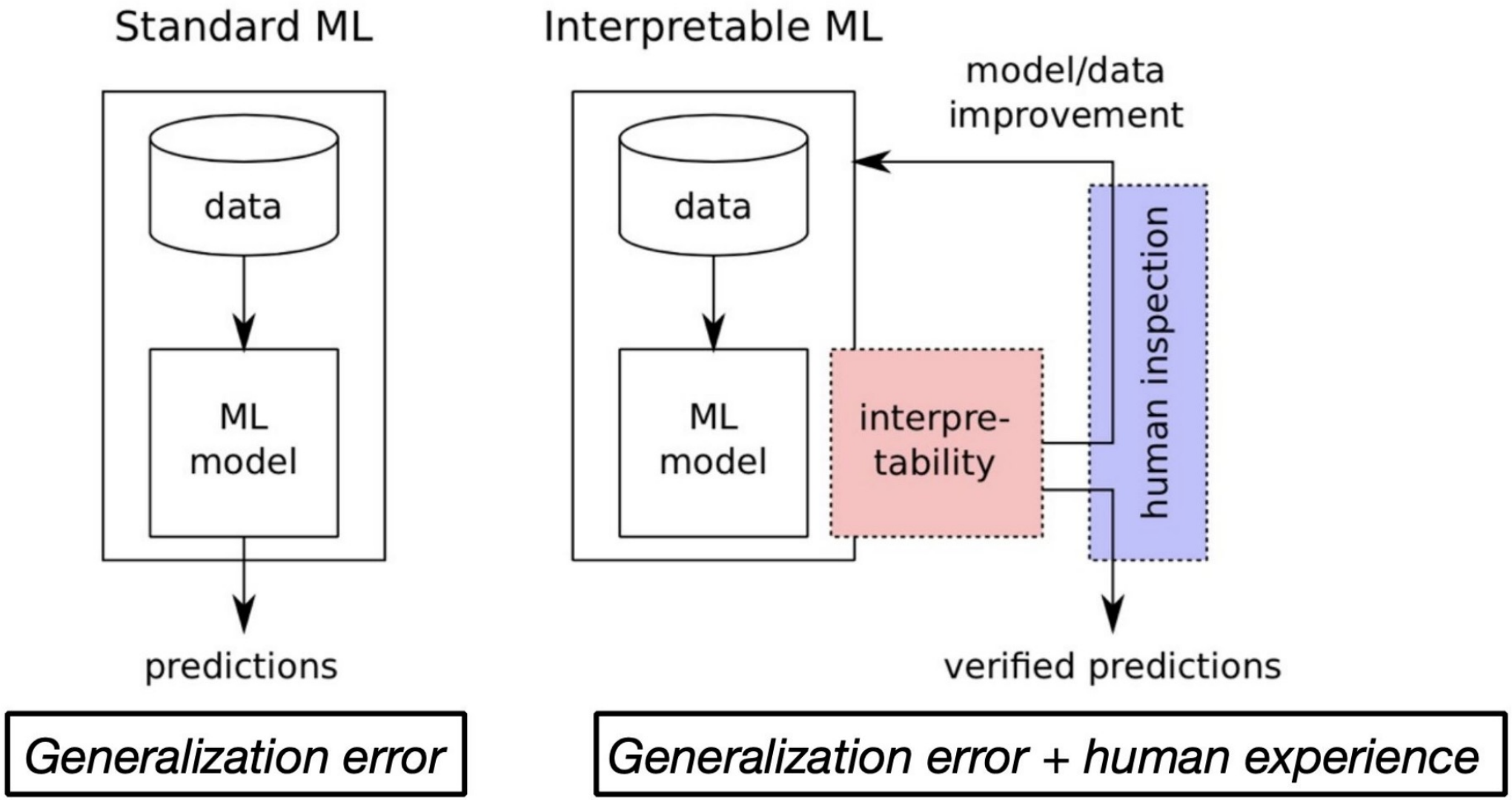
The AI Black Box Problem



Trust and Accountability Debugging and Improving Models Regulatory Compliance

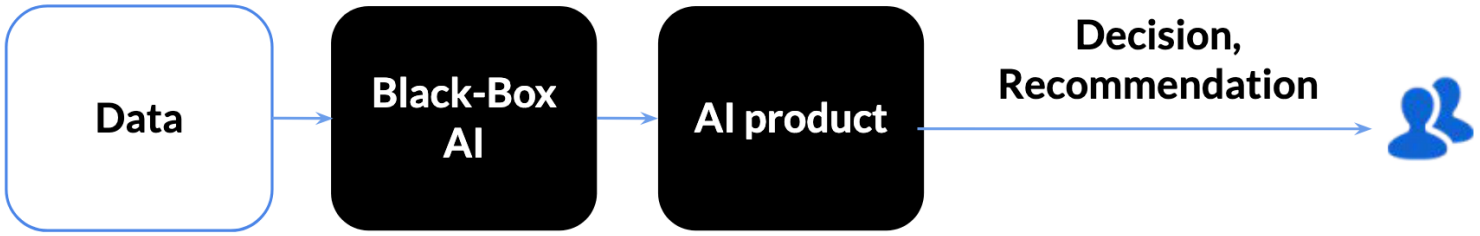


Why Explainability: Improve ML Model



Why Explainability: Improve ML Model

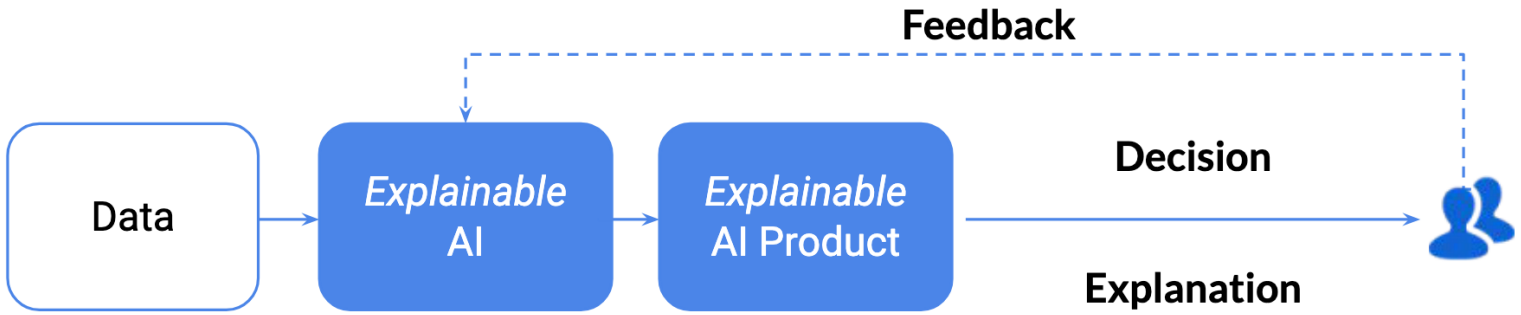
Black Box AI



Confusion with Today's AI Black Box

- Why did you do that?
- Why did you not do that?
- When do you succeed or fail?
- How do I correct an error?

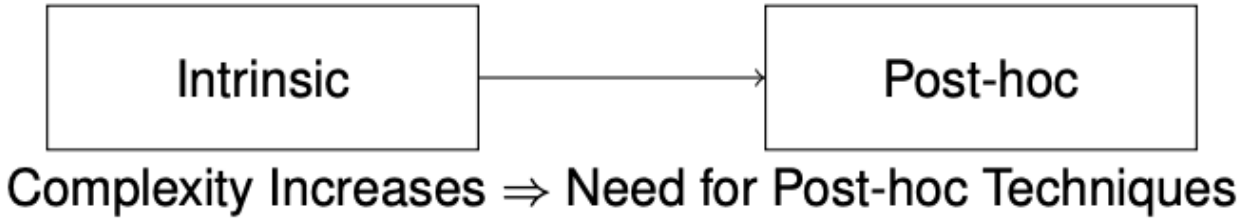
Explainable AI



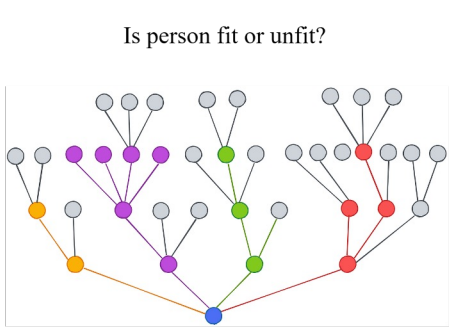
Clear & Transparent Predictions

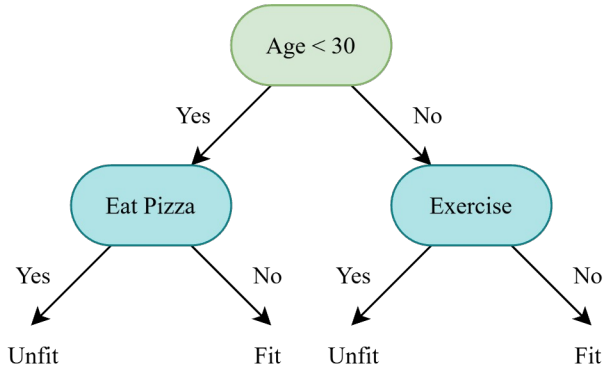
- I understand why
- I understand why not
- I know why you succeed or fail
- I understand, so I trust you

Intrinsic Interpretability vs. Post-hoc Interpretability



Intrinsic Interpretability

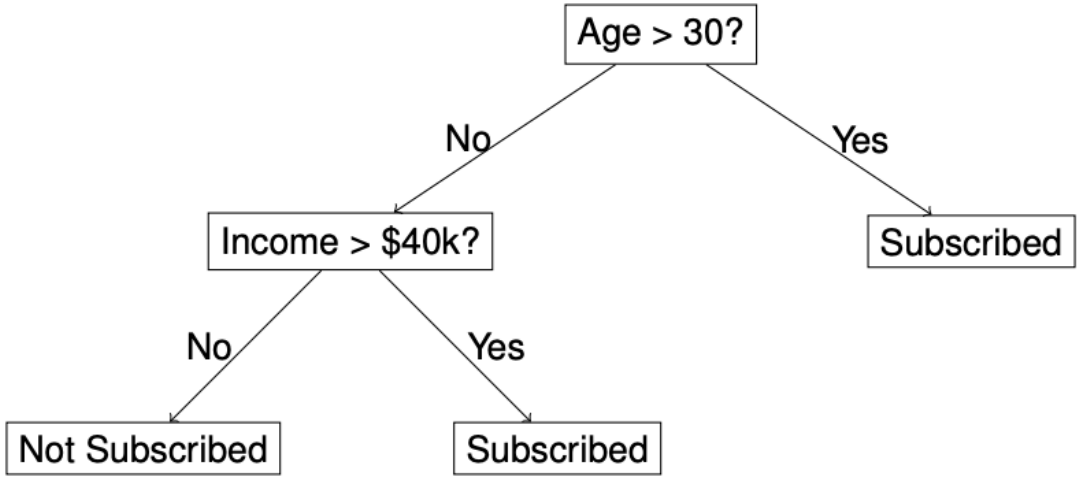


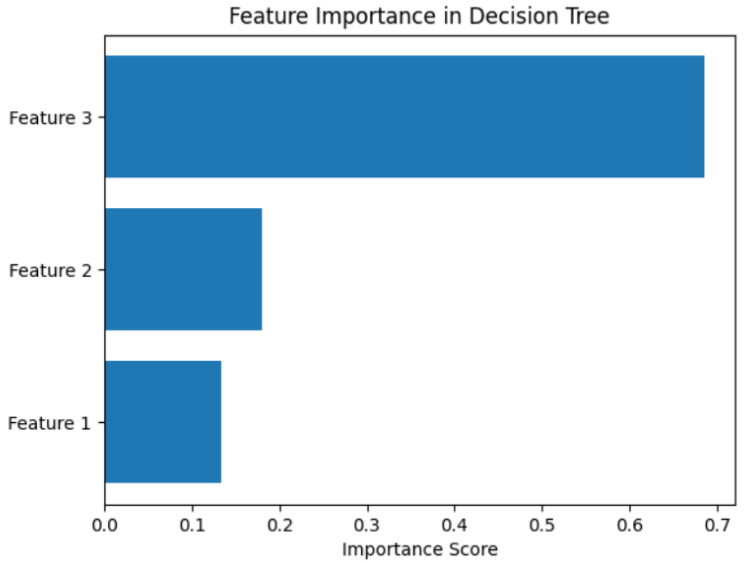


Post-hoc Interpretability

More complex models often require additional methods for interpretation after training

Decision Tree





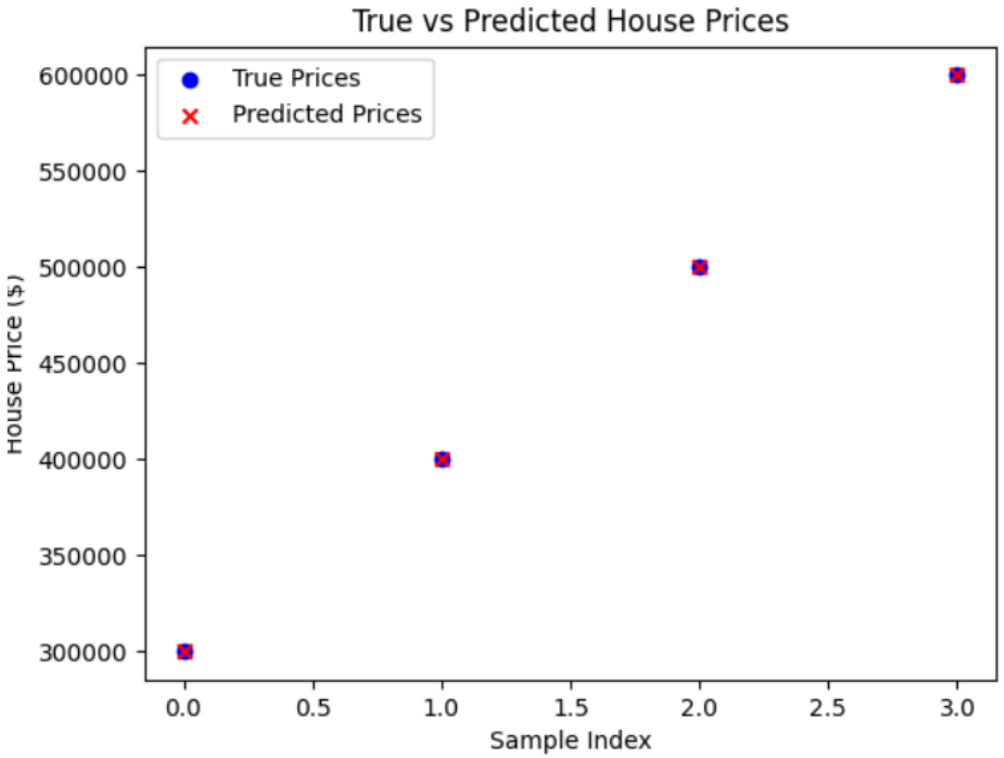

```python
from sklearn.tree import DecisionTreeClassifier

# Example of pre-pruning with maximum depth
clf = DecisionTreeClassifier(max_depth=4, min_samples_leaf=10)
clf.fit(X_train, y_train)
```

This structure highlights why decision trees are considered interpretable: every decision

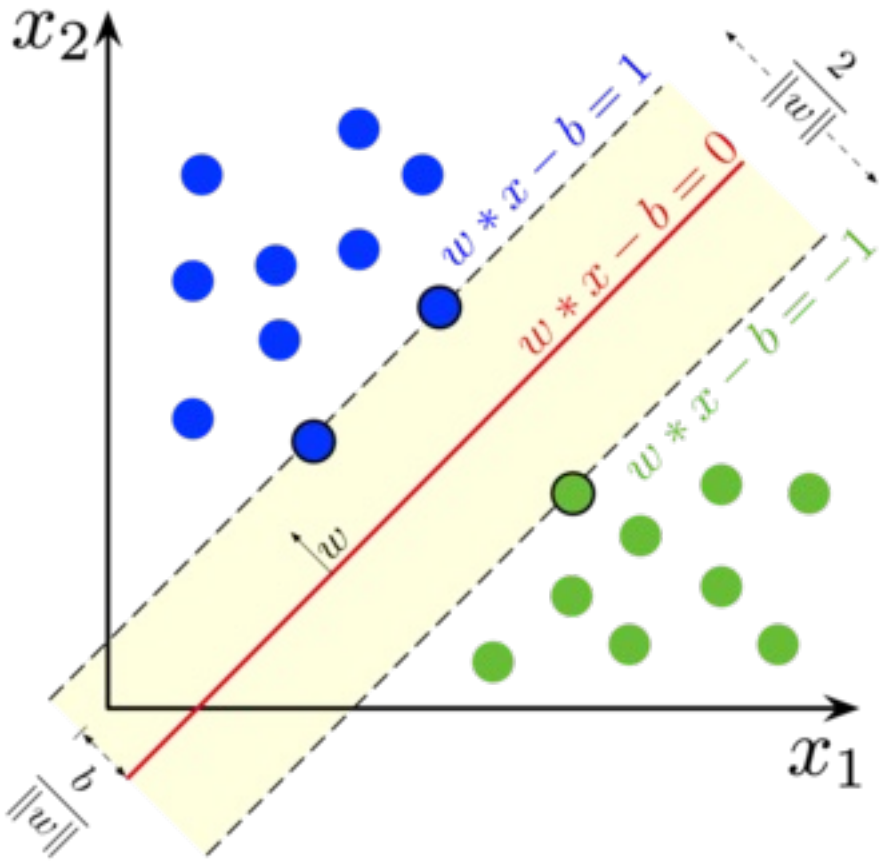
Linear Models

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$



A linear regression is simple and interpretable,

Interpretability of Support Vector Machines



$$\mathbf{w} \cdot \mathbf{x} + b = 0,$$

$$\mathbf{w} \cdot \mathbf{x_i} + b = \pm 1,$$

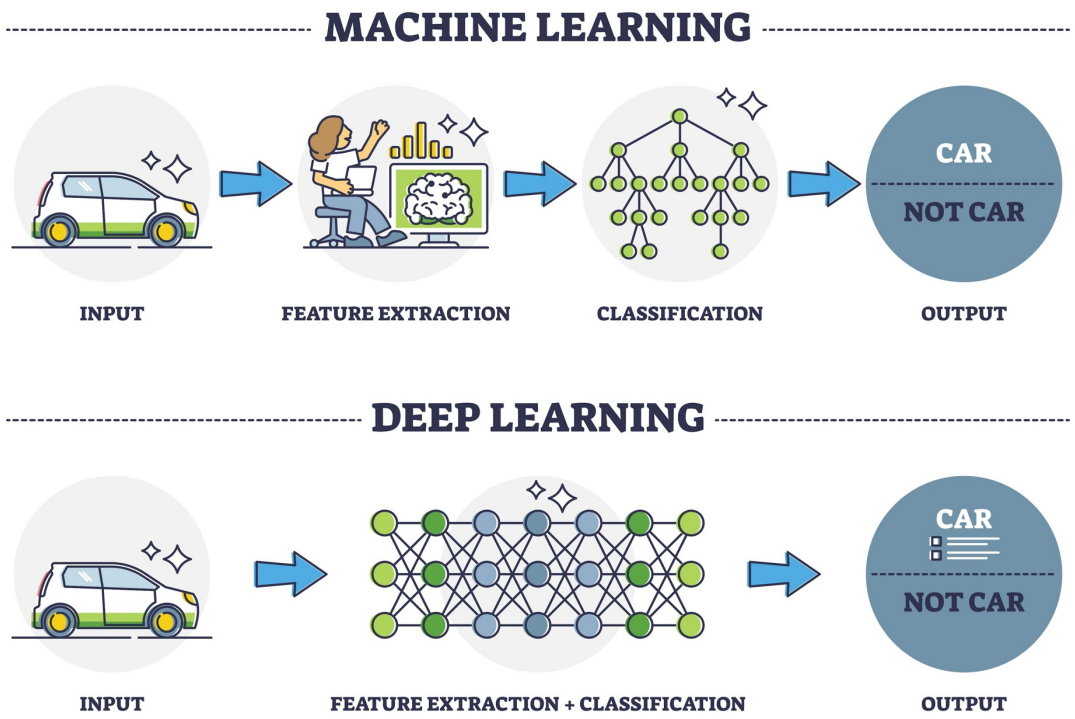
In the case of a linear kernel, the decision boundary is straightforward and interpretable, especially in low-dimensional spaces.

Interpretability is significantly reduced when using non-linear kernels (e.g., RBF), as the decision boundary becomes complex and difficult to visualize

Why Are Deep Learning Models Hard to Interpret

High Complexity of the Model

Non-linearity and Feature Abstraction

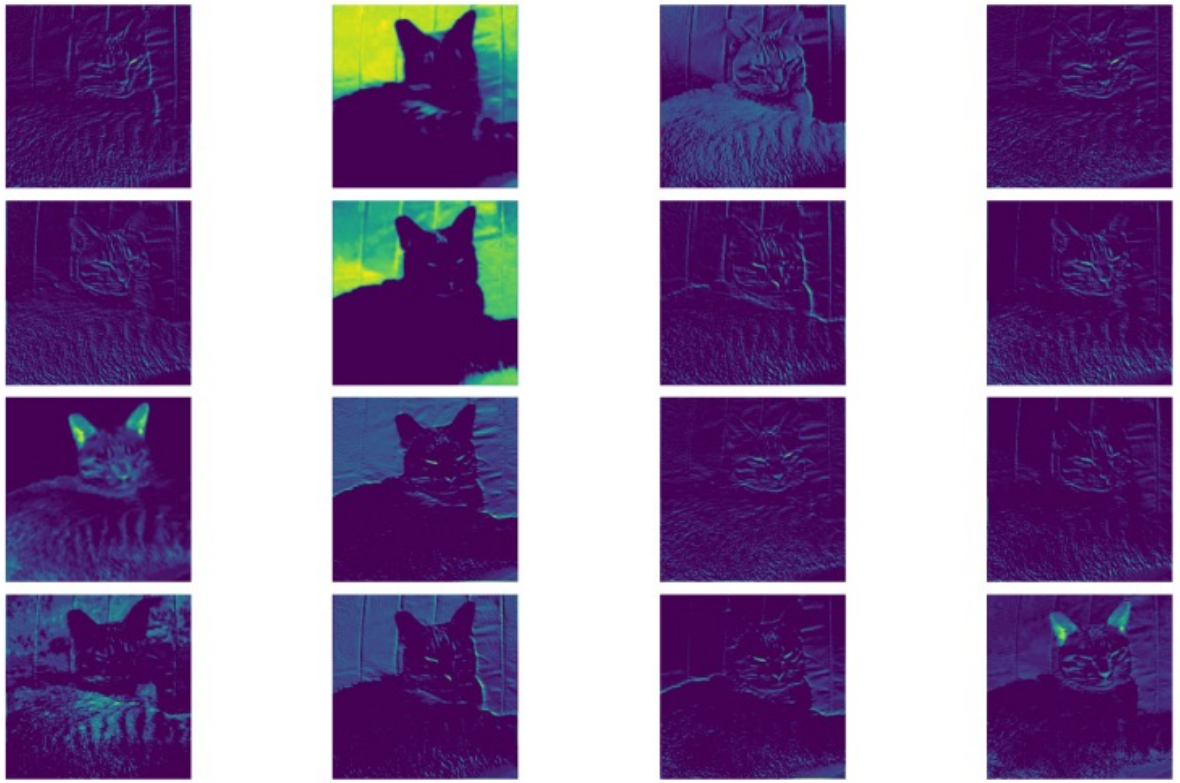


Lack of Explicit Structure

The Curse of Dimensionality

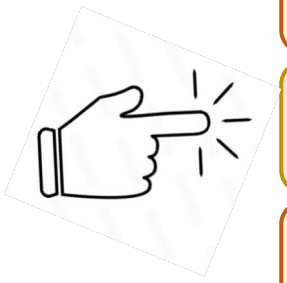
Interpretability of CNNs

Original Cat Image



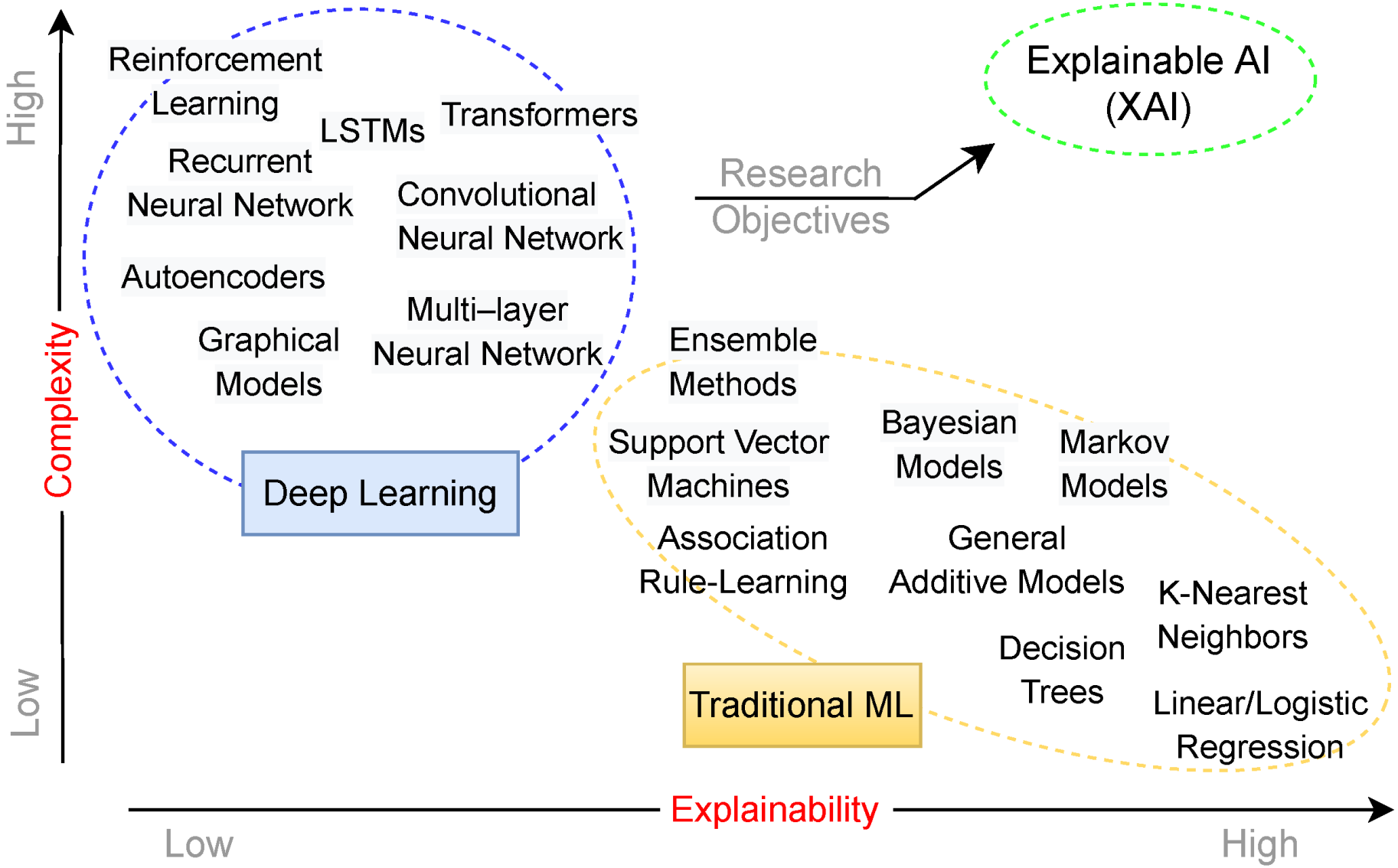
Feature Maps from the First Convolutional Layer of VGG16

Outline

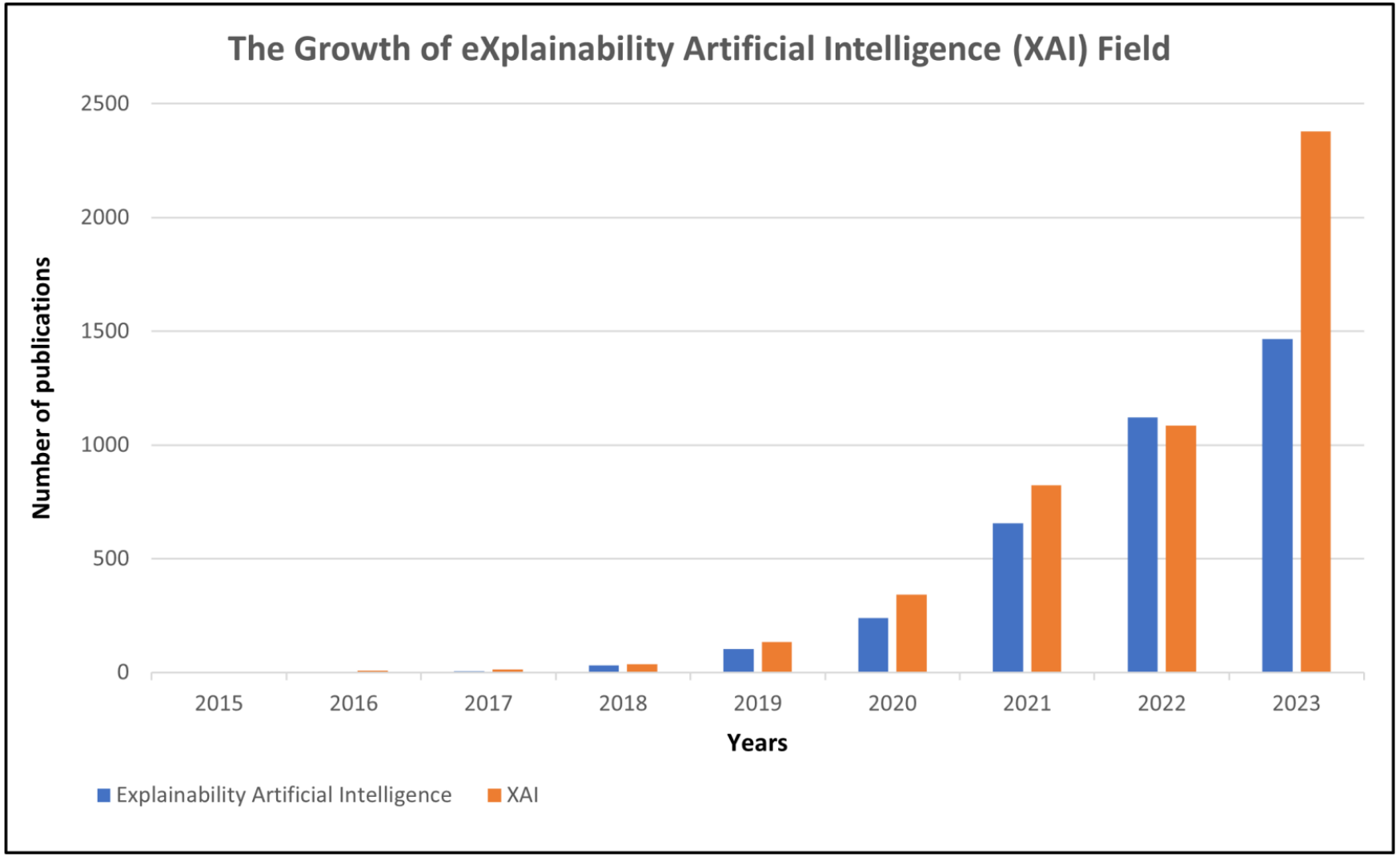


- Why is Interpretability Need?
- Evolution of XAI
- LIME for AI Explainable
- LIME: Example

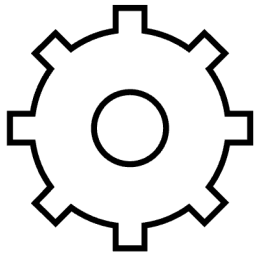
Evolution of XAI



Evolution of XAI



Evolution of XAI



Early Symbolic Systems

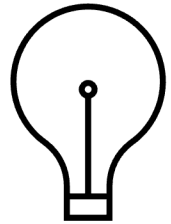
Initial AI systems (e.g., expert systems) had built-in transparency due to rule-based architectures.

(1950s–1980s)

Black-Box Challenges

The rise of neural networks and deep learning obscured model logic, prompting the need for interpretability tools.

(1980s/1990s–Present)

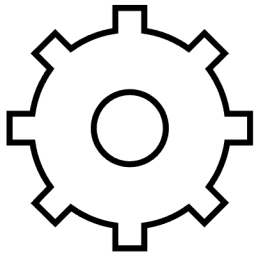


Emergence of XAI Methods

Techniques like LIME, SHAP, and saliency maps emerged post-2016 to restore insight into complex model behavior.

(2016–Present)

Evolution of XAI



Early Symbolic Systems

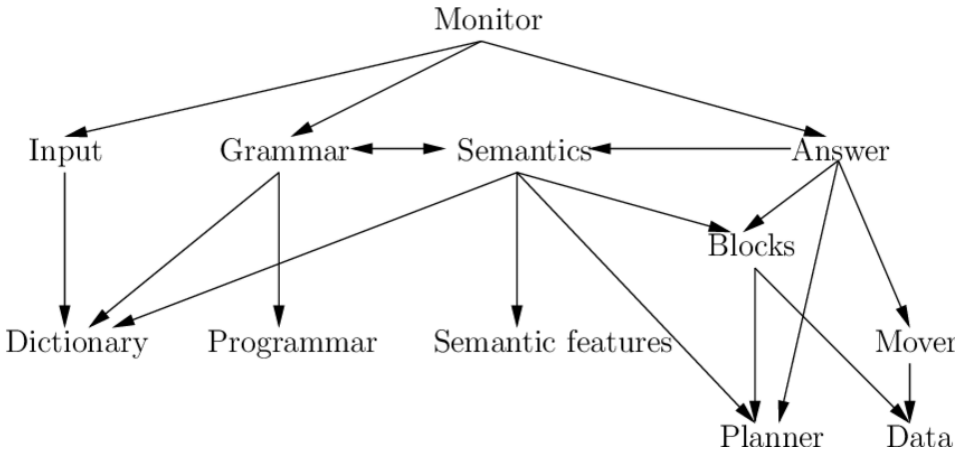
Initial AI systems (e.g., expert systems) had built-in transparency due to rule-based architectures.

Transparency by Design

Human-Centric Reasoning

Inflexible and Low Performance

SHRDLU model



Evolution of XAI

Black-Box Challenges

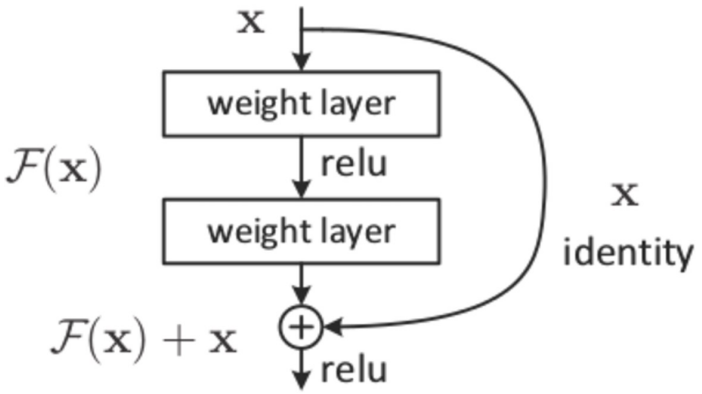
The rise of neural networks and deep learning obscured model logic, prompting the need for interpretability tools.

Adapt to Real World Problem

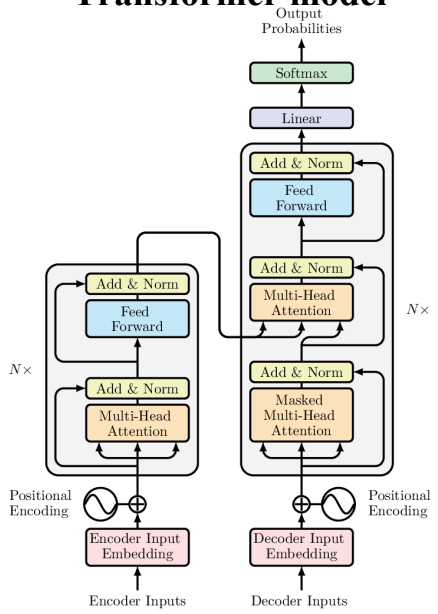
Better Performance

Opacity in Decision-Making

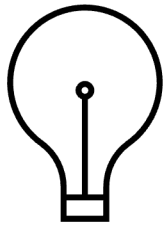
ResNet model



Transformer model



Evolution of XAI



Emergence of XAI Methods

Techniques like LIME, SHAP, and saliency maps emerged post-2016 to restore insight into complex model behavior.

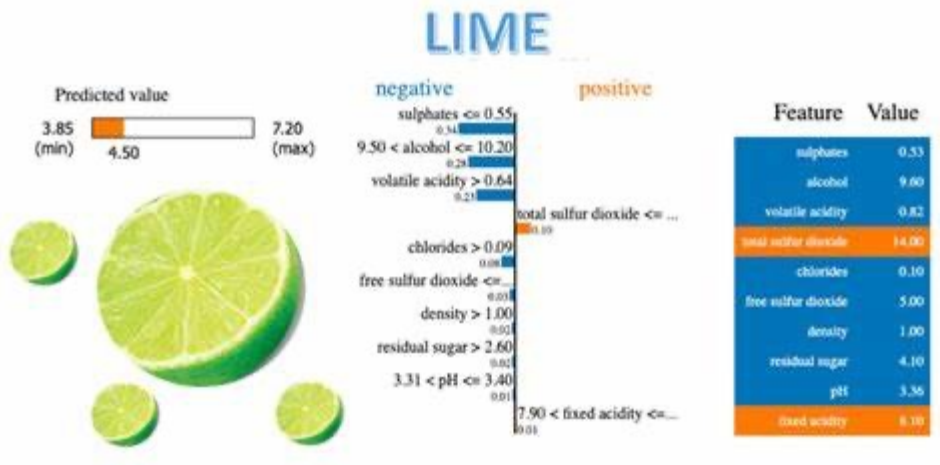
Model-Agnostic Techniques

LIME, SHAP, ANCHOR

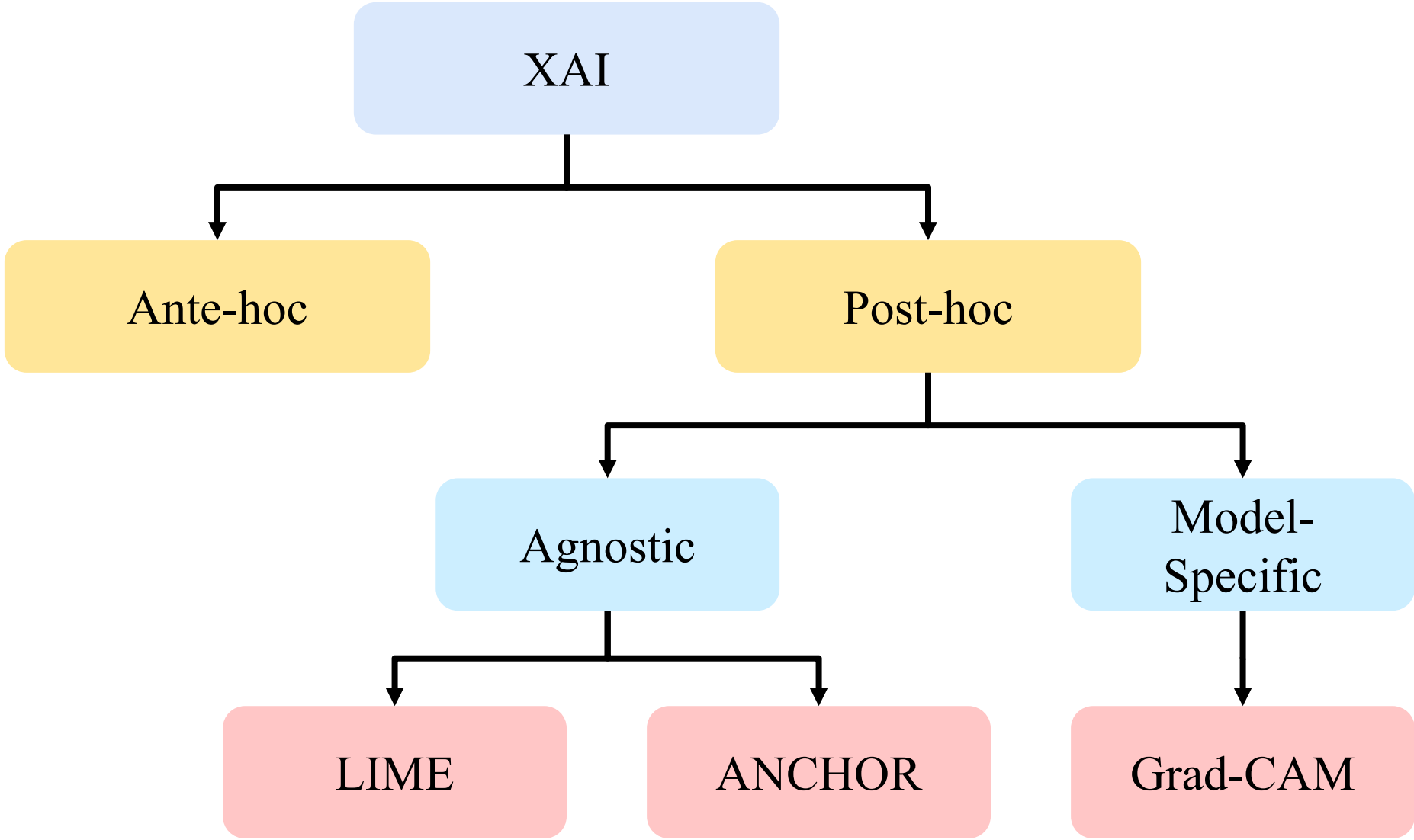
Visualization-Based Techniques

Grad-CAM

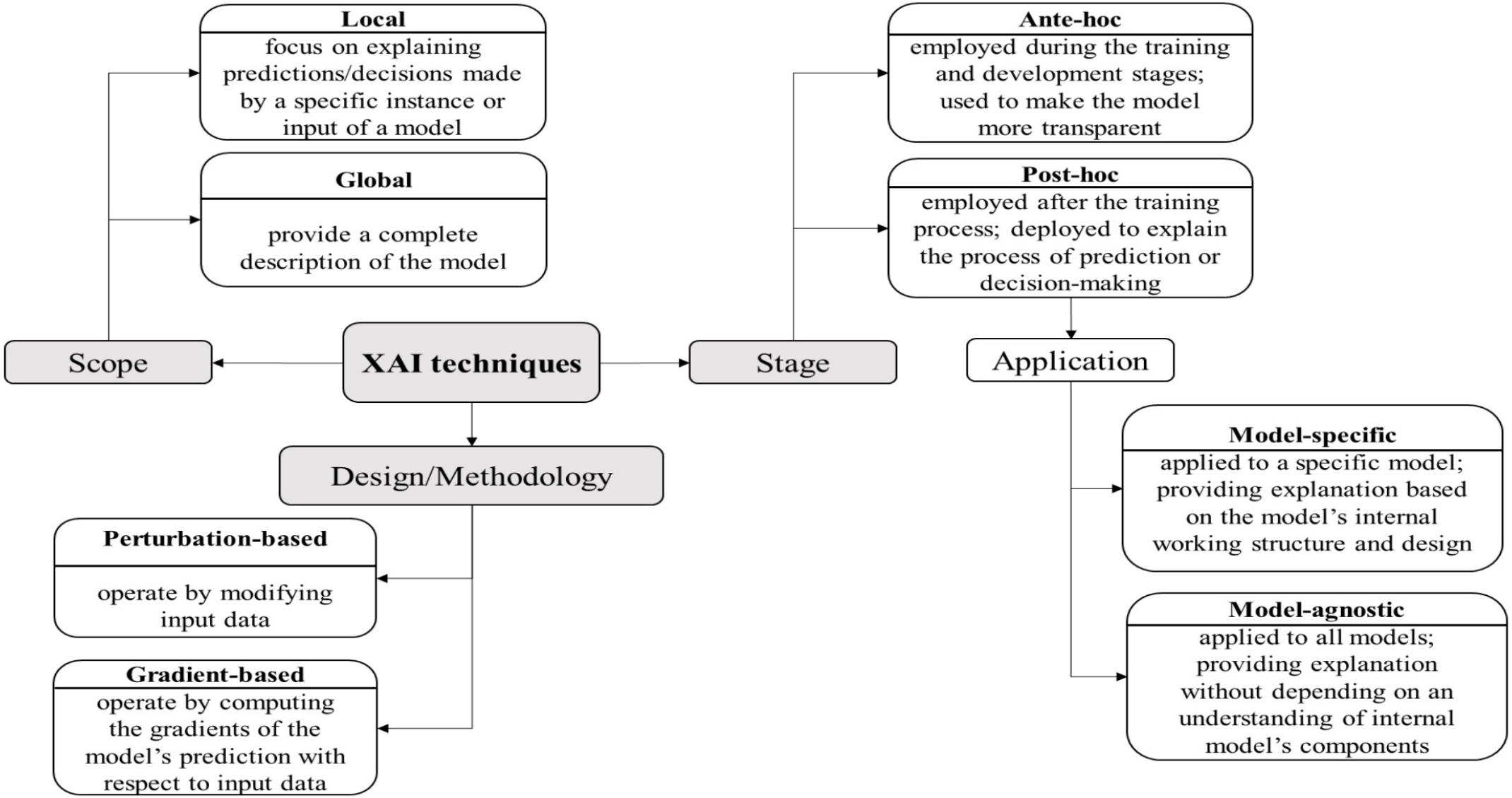
Integrated Gradients Techniques



XAI Taxonomy



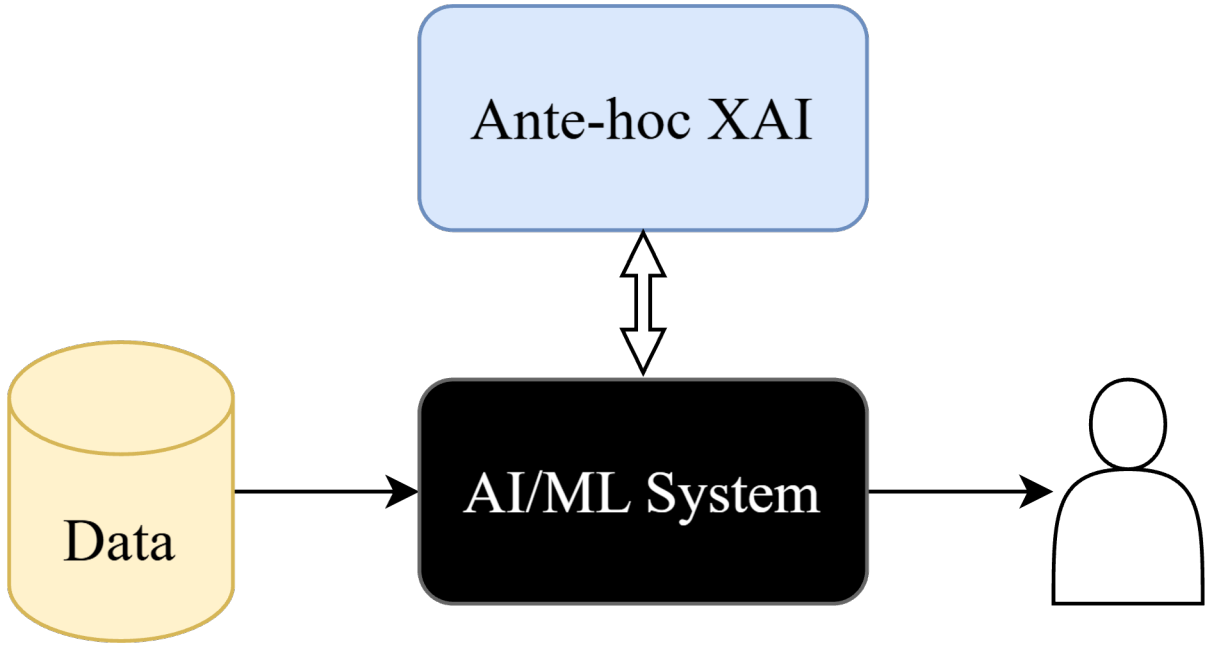
XAI Taxonomy



XAI Taxonomy

Ante-hoc

- Interpretable by nature
- Simplicity
- Transparency



Decision Trees
Linear Regression
Bayesian Rule Lists
Attention Mechanisms

Attention is a special case of XAI. Its don't give causal explanation of the model's decision.

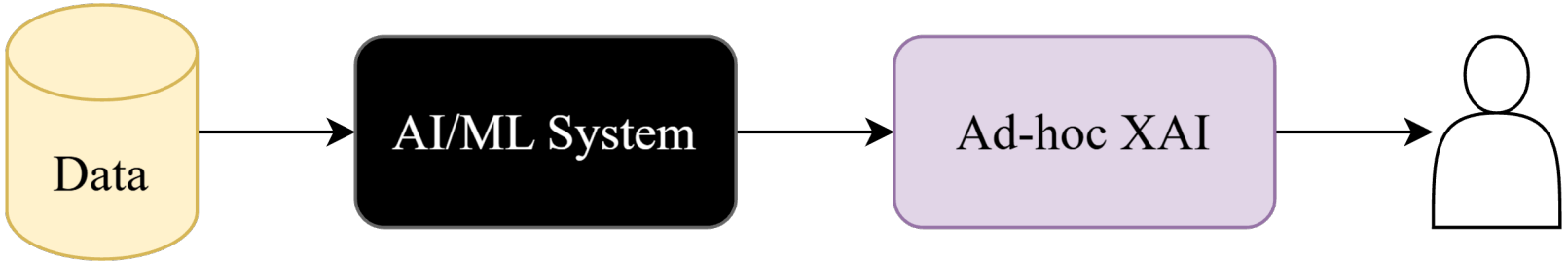
XAI Taxonomy

Post-hoc

- Explanatory Layer
- Approximation

Provide an external explanation without changing the model's original structure or performance.

Approximation of the internal logic, which may not capture all nuances.



LIME
SHAP
ANCHOR

Attention as Explanation

Attention heatmap of negative review

observed model

After 15 minutes watching the movie i was **asking** myself what to do leave the theater sleep or try to keep watching the movie to see if there was anything worth i finally watched the movie what a **waste** of time maybe i am not a 5 years old kid anymore

$P\ (negative) = 99\ \%$

$\neq$

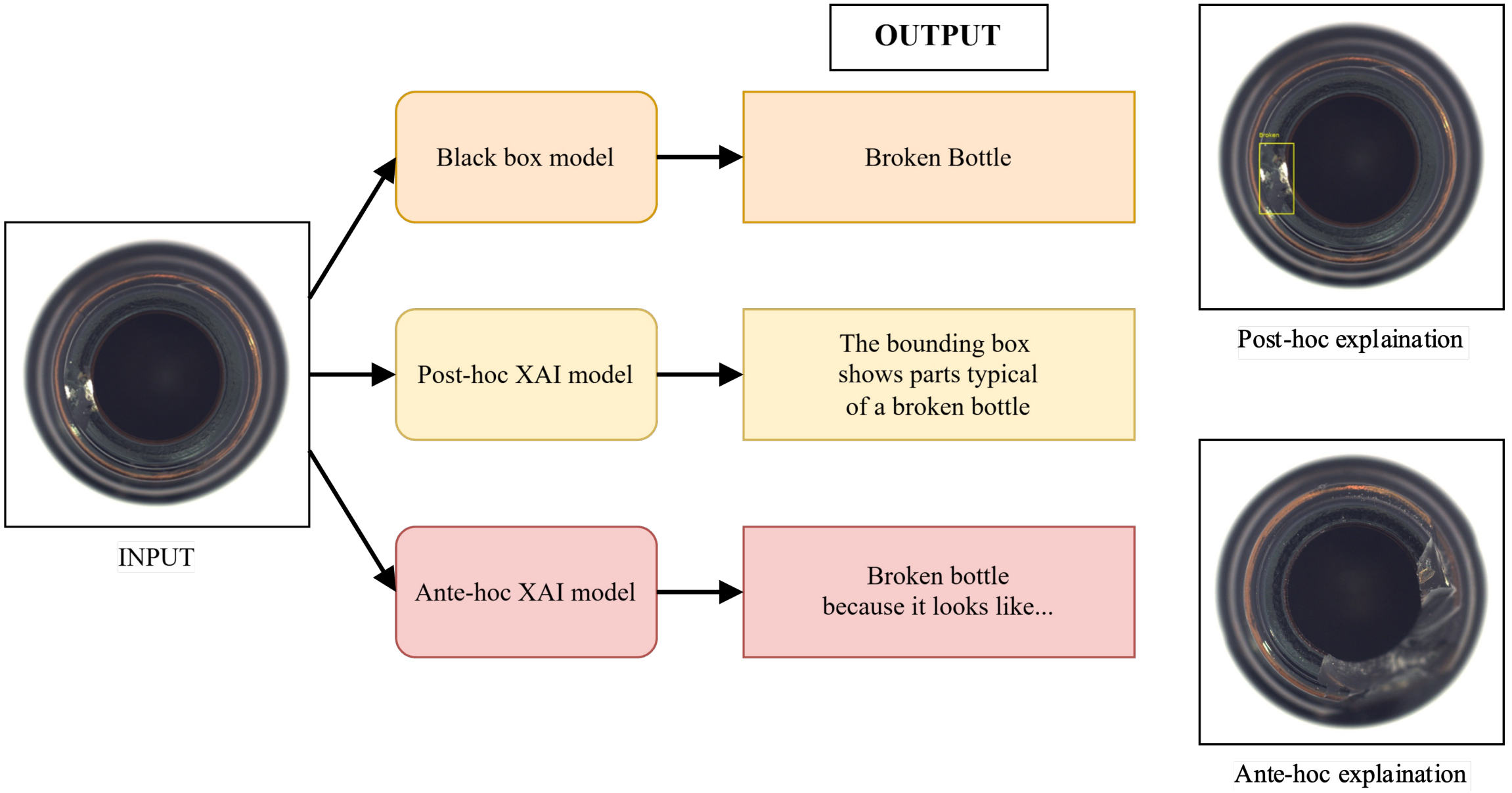
adversarial model

After 15 minutes watching the movie i was asking **myself** what to do leave the theater sleep or try to keep watching the movie to see if there **was** anything worth i finally watched the movie what a waste of time maybe i am not a 5 years old kid anymore

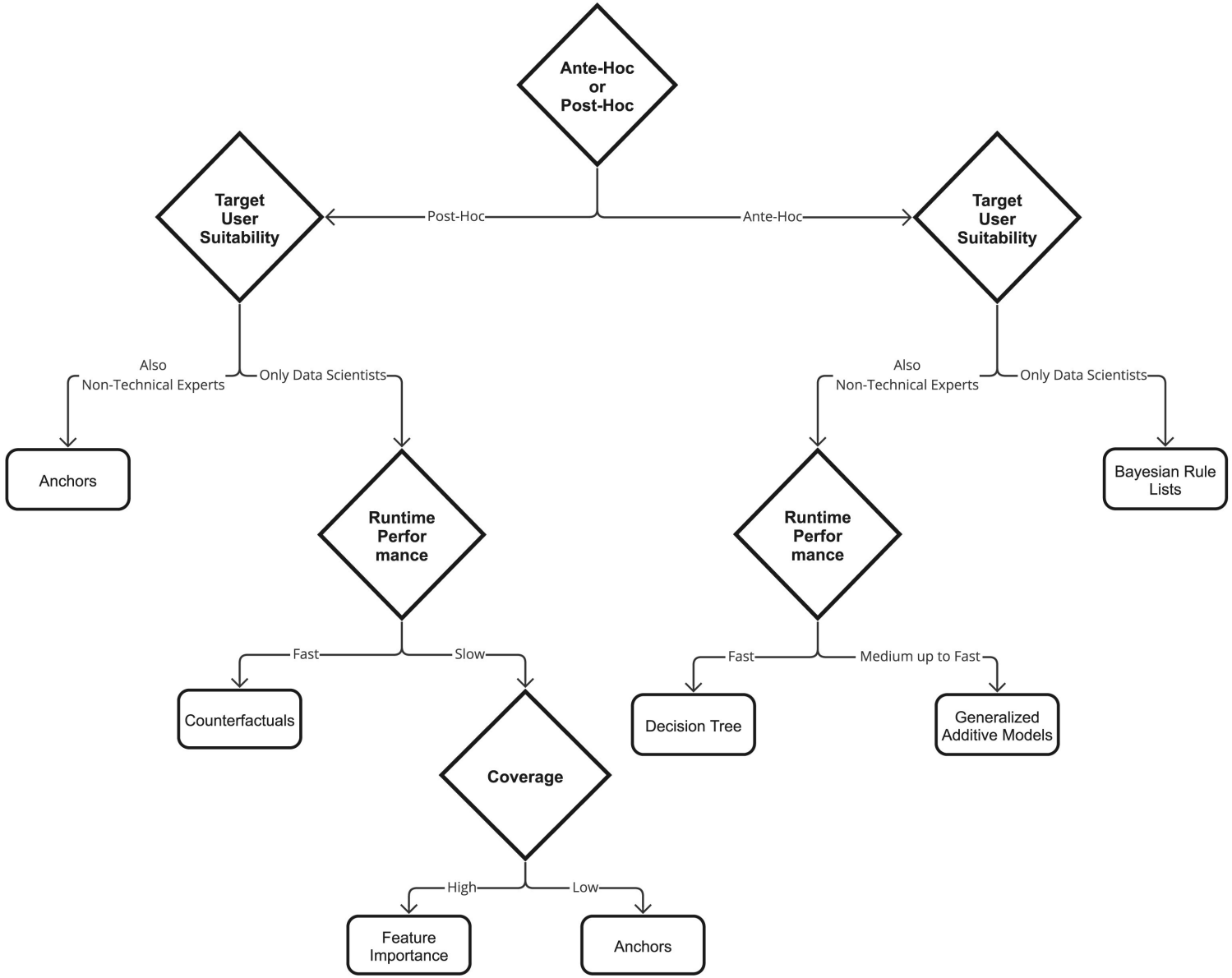
$=$

$P\ (negative) = 99\ \%$

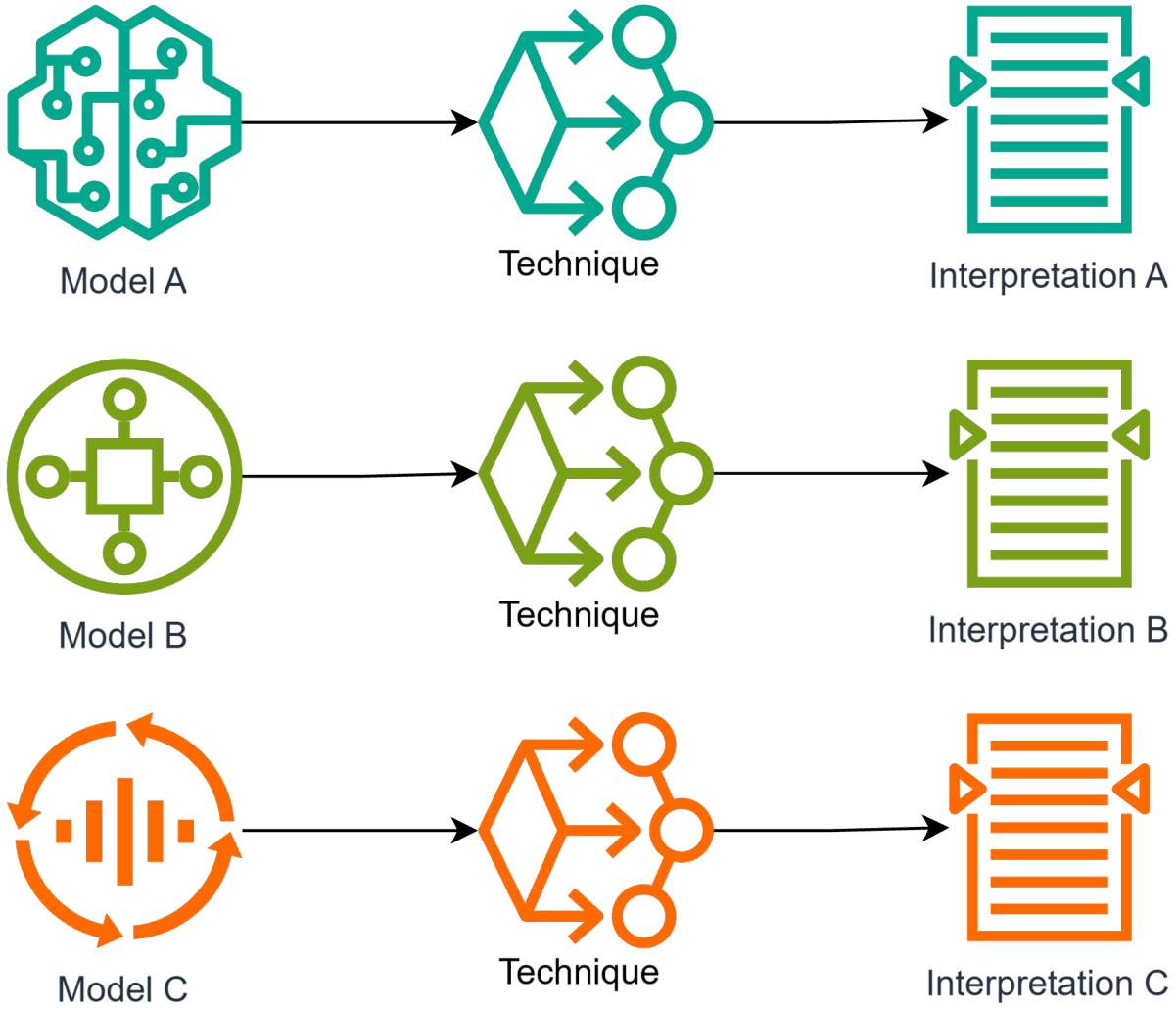
Ante or Post

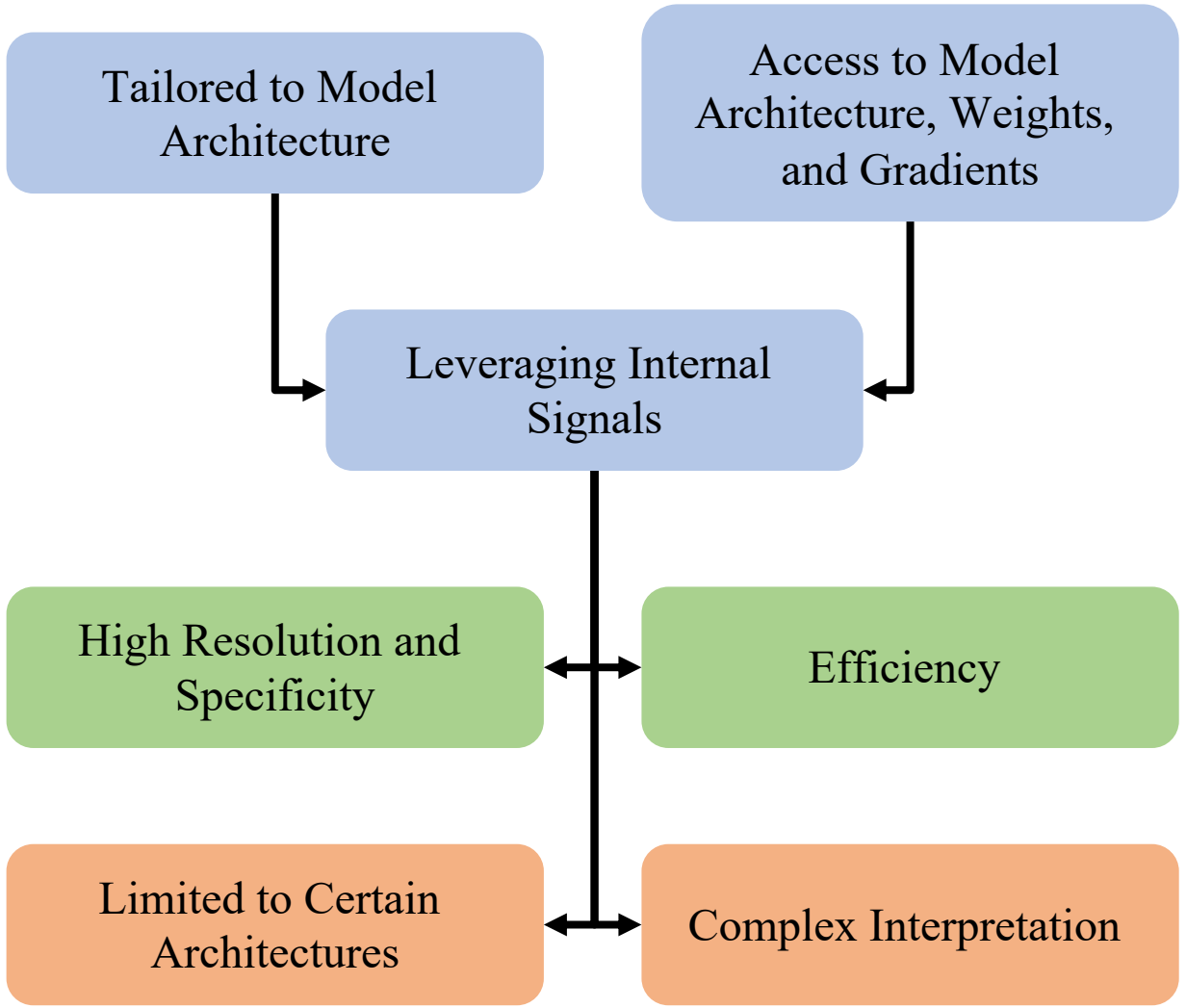


Ante or Post

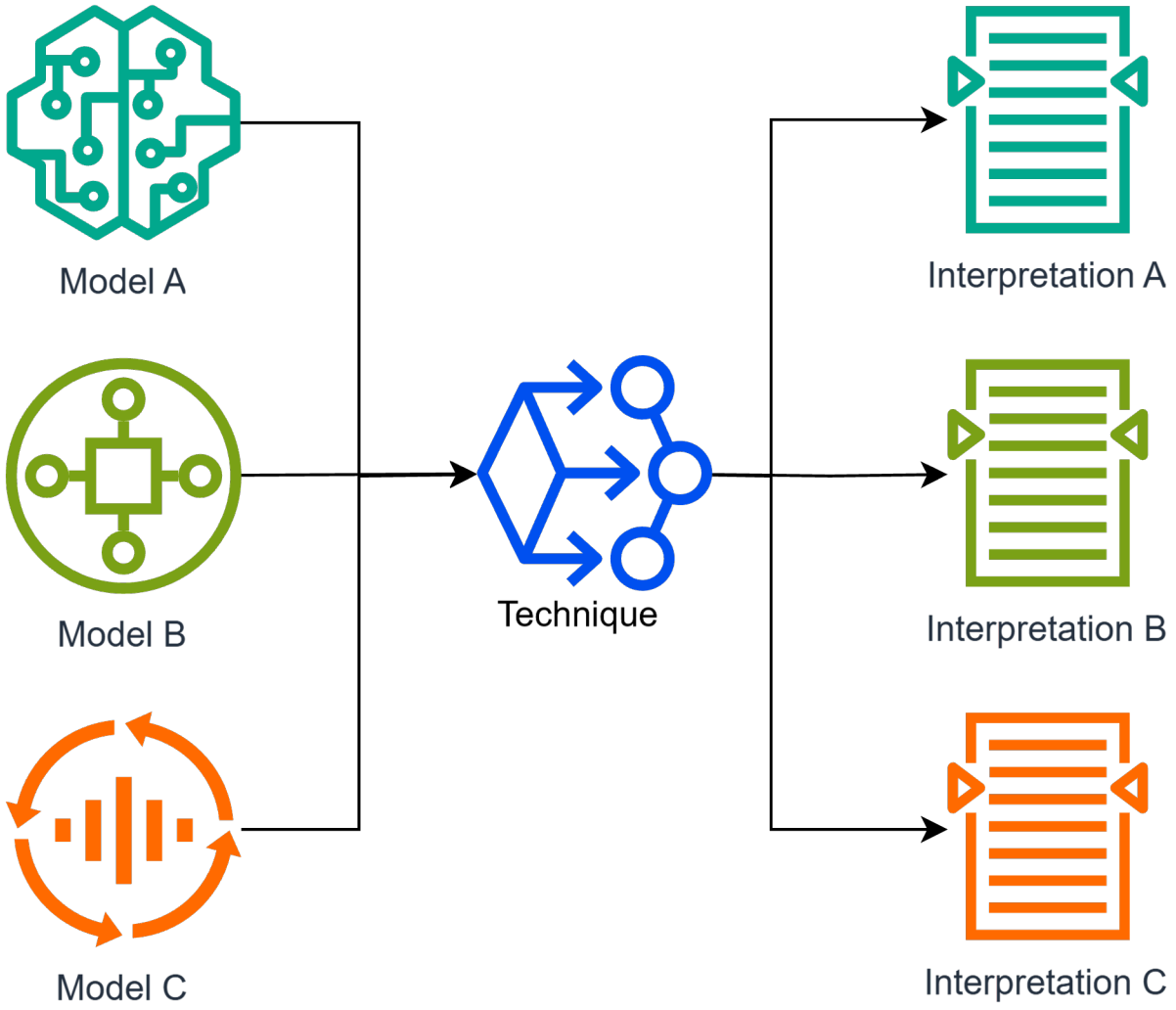


Model-Specific





Model-Agnostic



Perturbation−Based Analysis

Analys Different Between Input and Output

Versatility

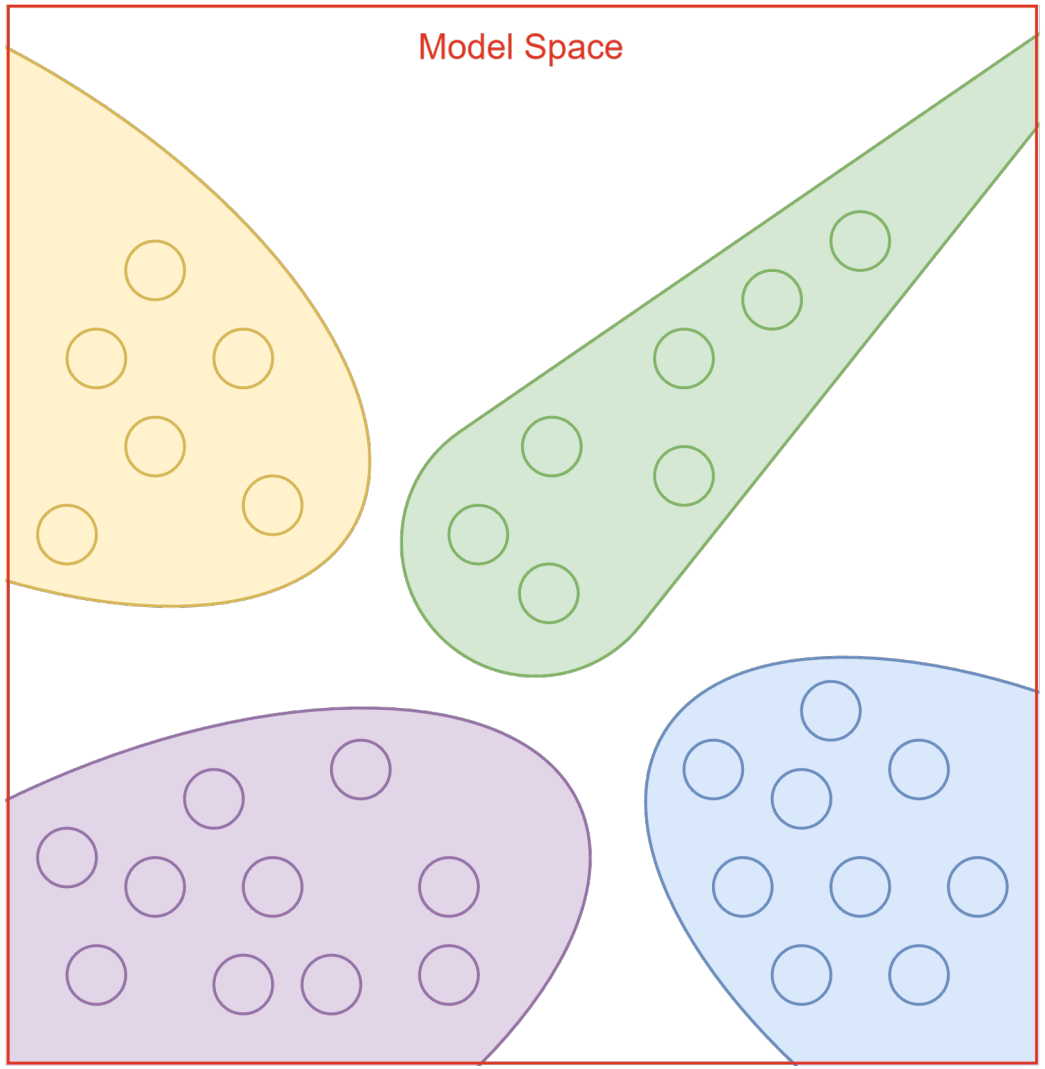
User−Friendly

Computational Cost

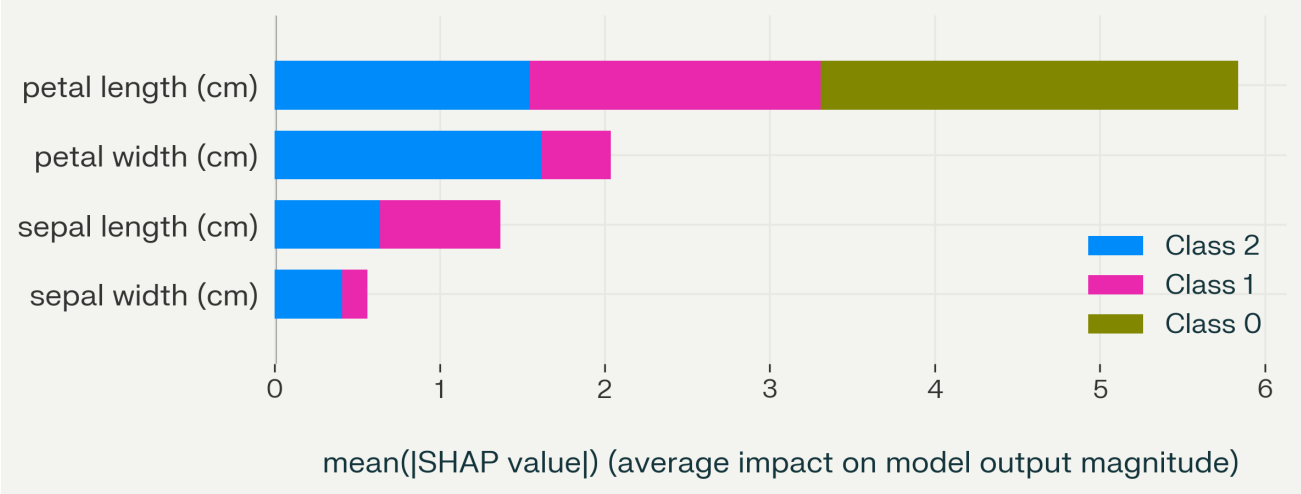
Approximation Quality

AI VIET NAM
@aivietnam.edu.vn

Global Explanations



What are the key factors driving the model's overall decisions?



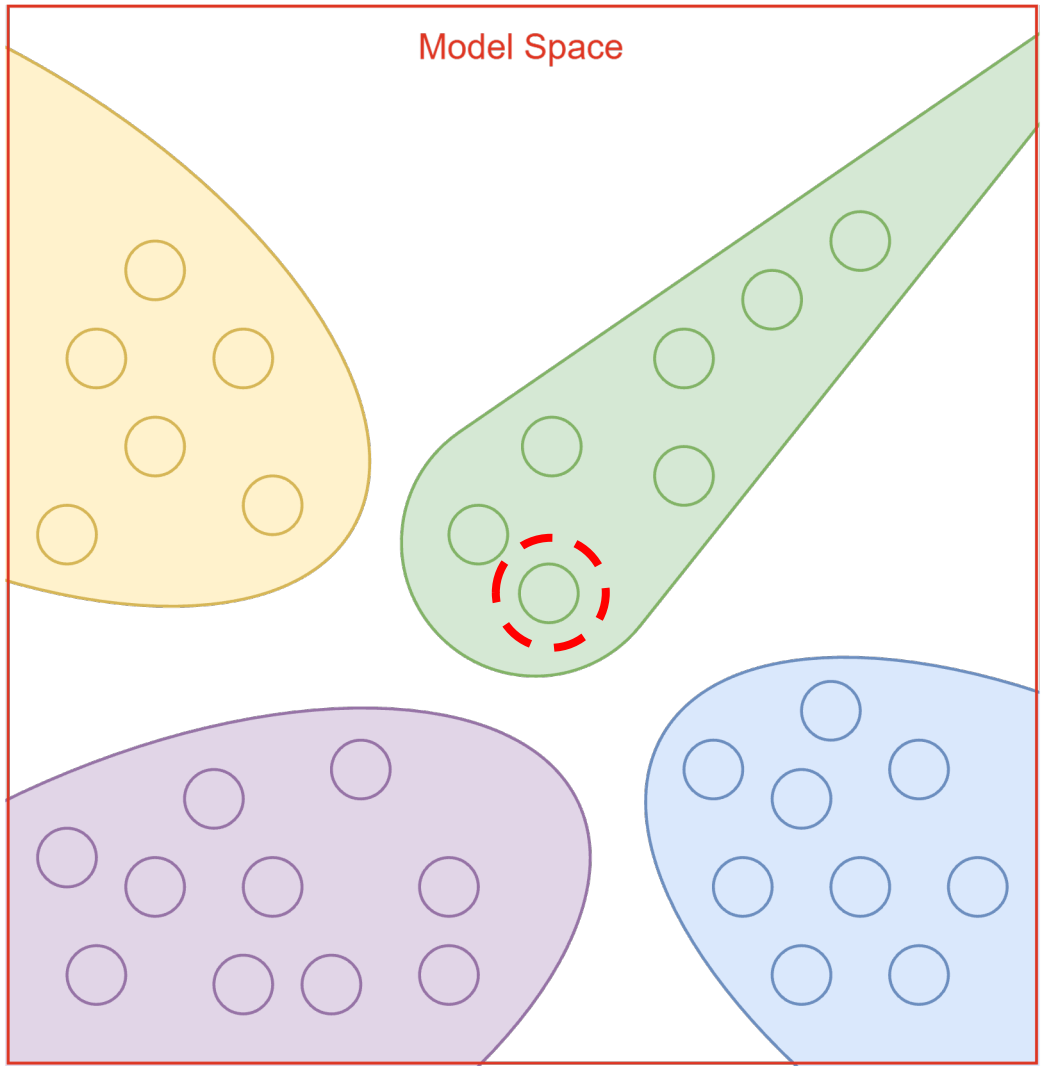
Simplified Communication

Comprehensive Insight

Approximation Errors

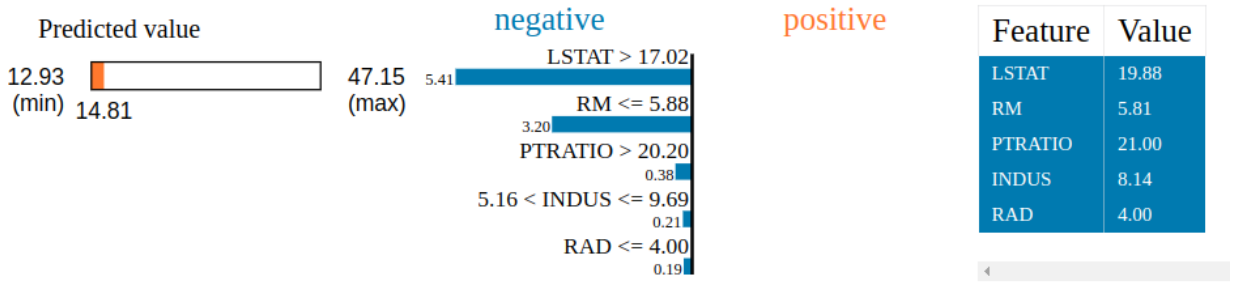
AI VIET NAM
@aivietnam.edu.vn

Local Explanations



Why did the model make this decision for this specific case?

Giving an instant of a review. Why model predict negative in this case?



Precision

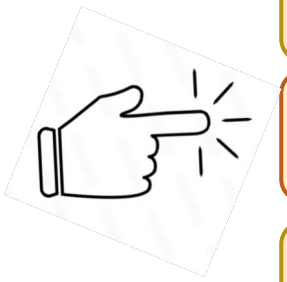
Trust

Fidelity

Limited Scope

Undifined "local"

Outline

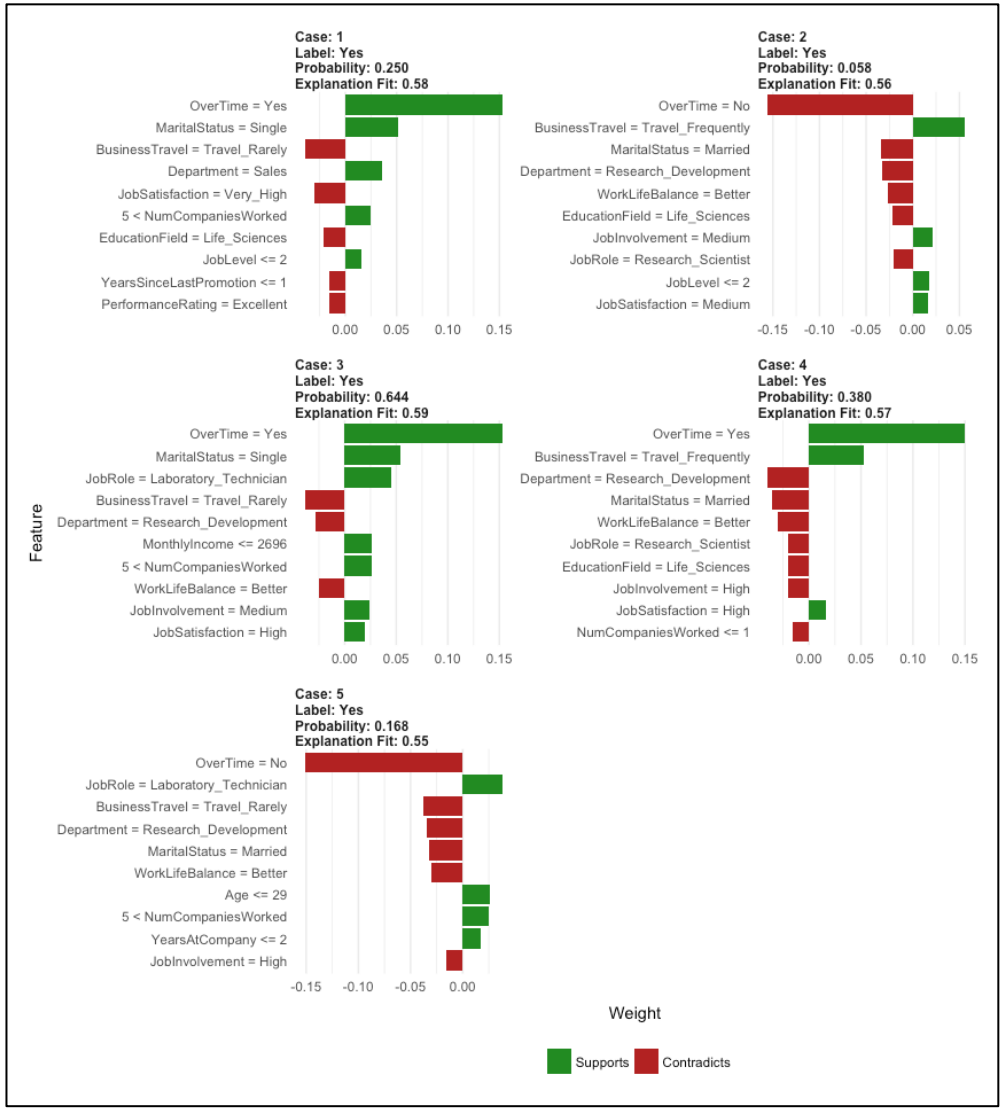


- Why is Interpretability Need?
- Evolution of XAI
- LIME for AI Explainable
- LIME: Example

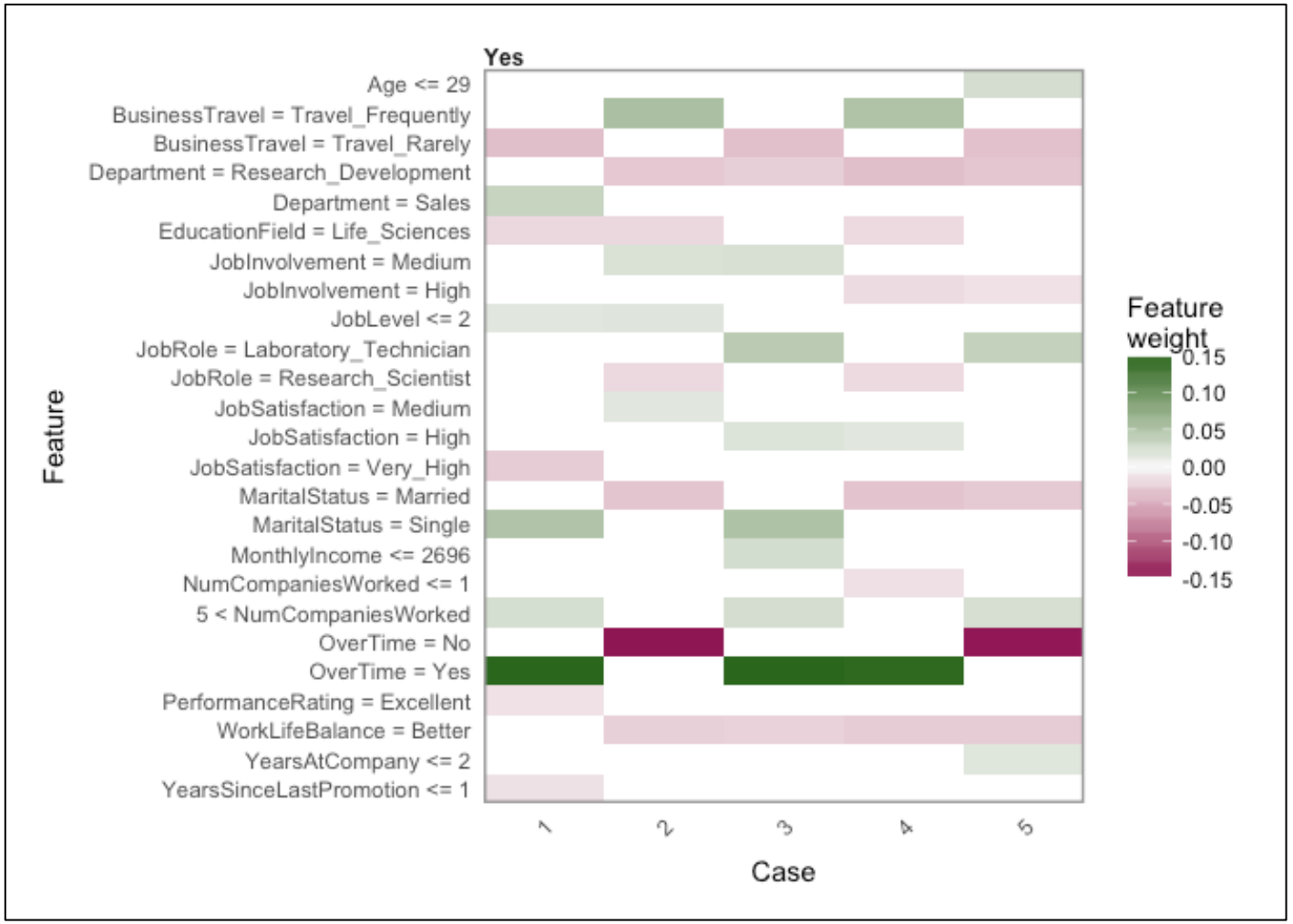
AI VIET NAM
@aivietnam.edu.vn

LIME Explanations

LIME local explaination



LIME global explaination



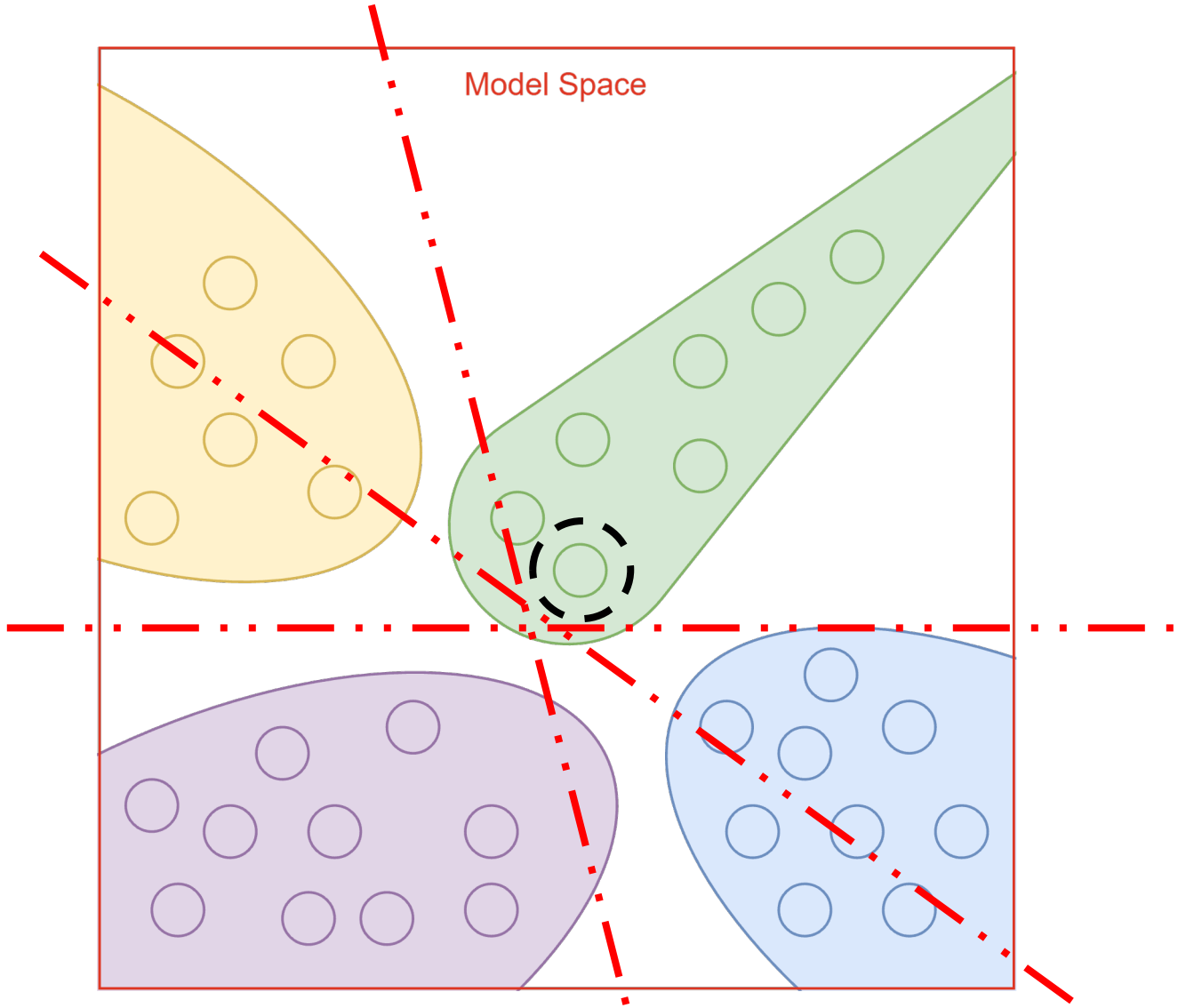
Local Interpretable Model-agnostic Explanations

LIME

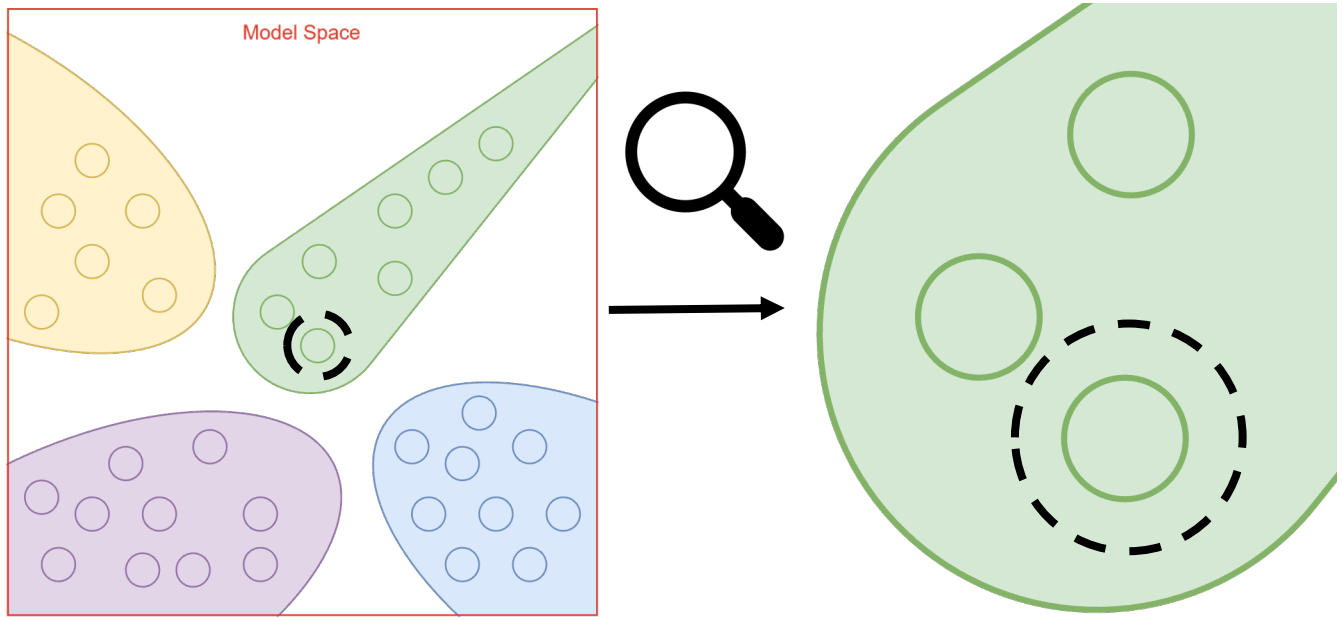
- Local Explanations
- Model−Agnostic
- Feature Importance Method
- Surrogate Models

What is the best line to split this instance?

How much we have zoom in to see the instance “local”?



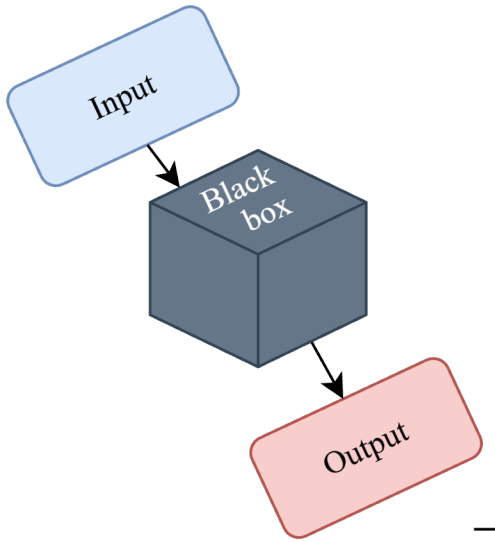
LIME

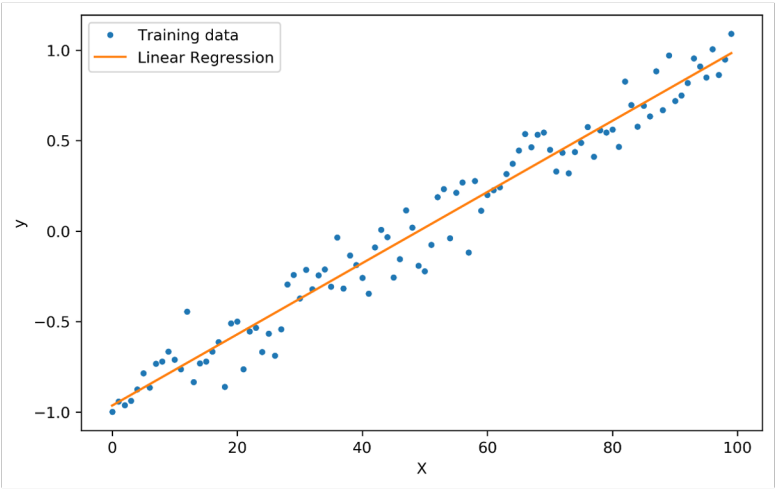


Zoom into the "local" zone

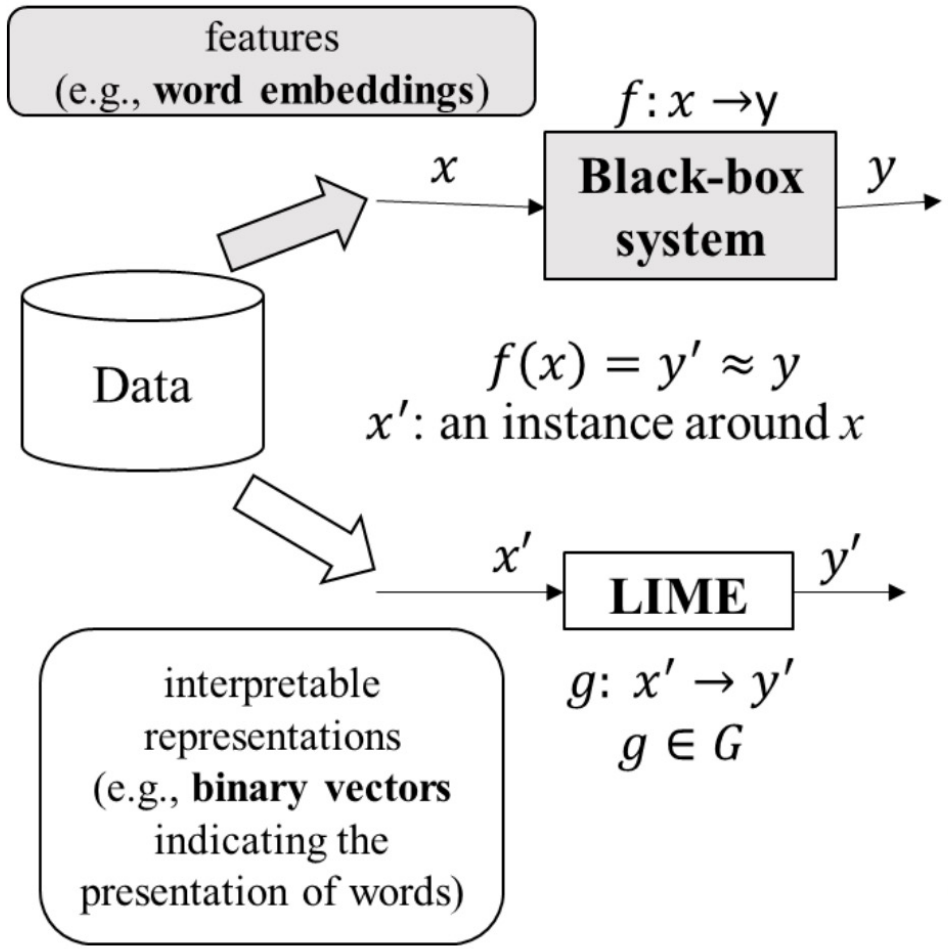
Better approximation for white-box model

Using simple white-box model to mimics complex model





LIME



LIME XAI scope: Local

focusing on explaining predictions/decisions made by a specific instance or input of a model

an explanation of a prediction made by an instance x

model being explained

locality around x

$$\xi(x) = argmin_{g \in G}\ \zeta(f, g, \pi_x) + \Omega(g)$$

unfaithful measure of g

explanation model

complexity of g

https://arxiv.org/pdf/2409.00265

LIME Framework

measure of unfaithfulness | interpretable model | complexity measure

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \; \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

an instance

model to be explained

proximity measure to define locality

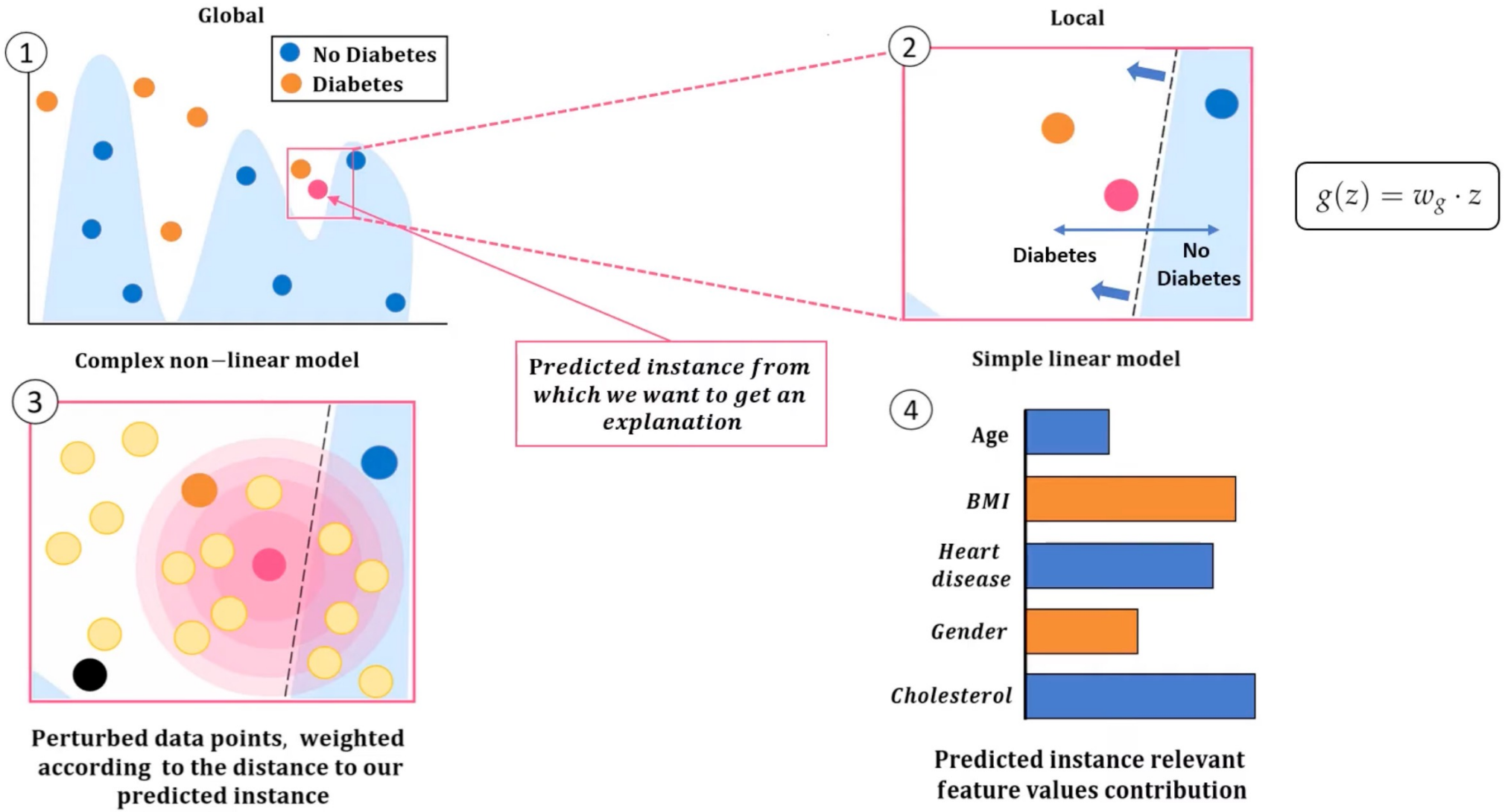
Limit interpretable model to become complex

Can be any function according to data type

Fidelity: How much the explanation **align** with model inner working.

Local: LIME weights samples near x more heavily so that g reflects local behavior.

LIME Framework



LIME Framework

$$x \in \mathbb{R}^d \xrightarrow[x = h_x(x')]{x' = h_x^{-1}(x)} x' \in \{0,1\}^{d'}$$

Original representation
the instance being explained

Human interpretable
representation
of this instance

Text

Binary vector indicating the presence or absence of a word.

	I	Love	You
I Love You	1	1	1
I You	1	0	1
-	0	0	0

Image

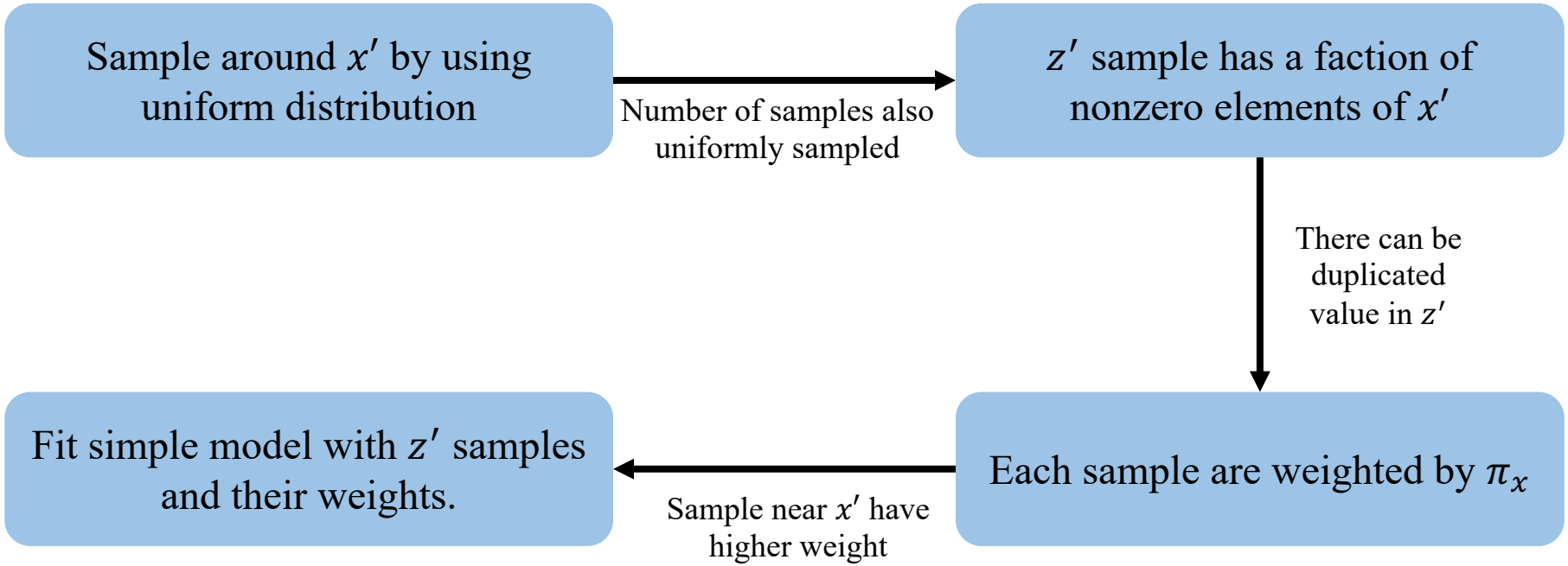
Binary vector indicating the presence or absence of a contiguous patch of similar pixel.

$$x' \in [1, 1, 1, 1]$$

LIME Framework

$$x \rightarrow x' \rightarrow z' \rightarrow z \rightarrow f(z)$$

$x \rightarrow x'$	$x' \rightarrow z'$	$z' \rightarrow z$	$z \rightarrow f(z)$
interpretable representation	sampling local area	inverse (interpretable representation)	obtain label



LIME Framework

$$x \rightarrow x' \rightarrow z' \rightarrow z \rightarrow f(z)$$

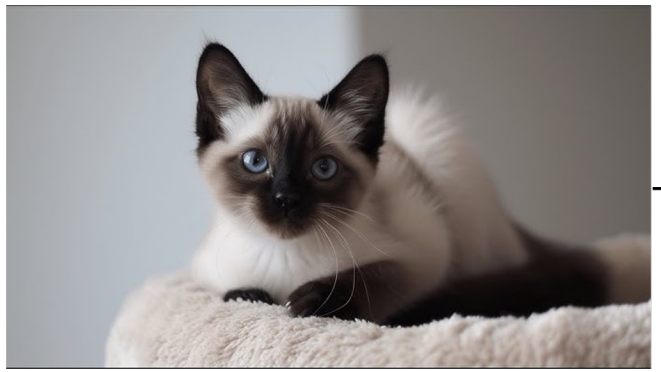
interpretable representation

sampling local area

inverse (interpretable representation)

obtain label

X X' Z' Z $F(Z)$



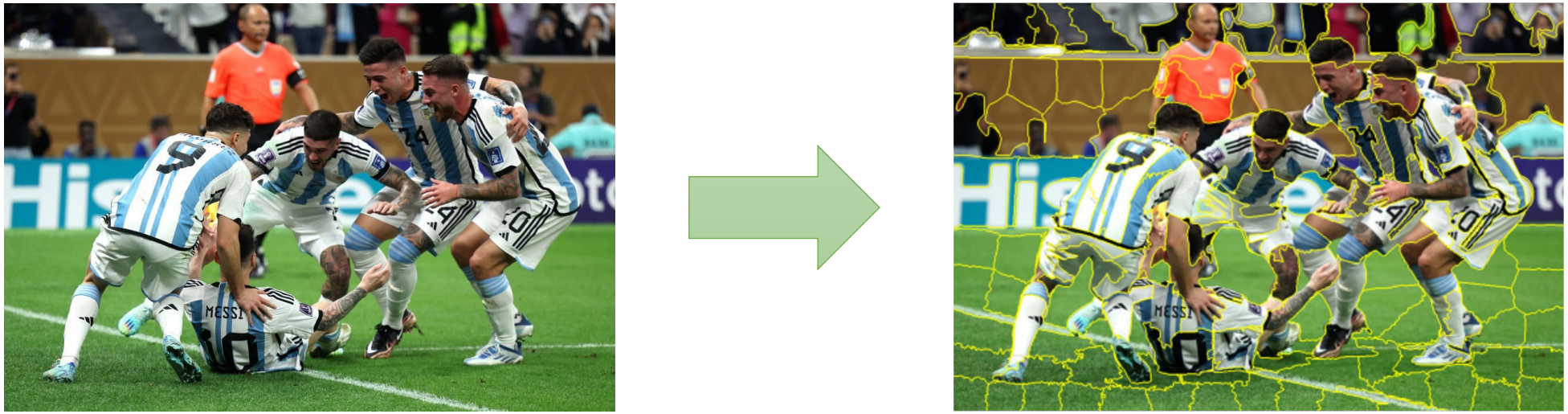
$$\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

'Siamese_cat', 0.929
'Egyptian_cat', 0.001
'washbasin', 0.001
'radio', 0.000
'streetcar', 0.000

Segmentation

❖ Image segmentation

Image segmentation refers to the division of an image into smaller regions according to certain predefined criteria.

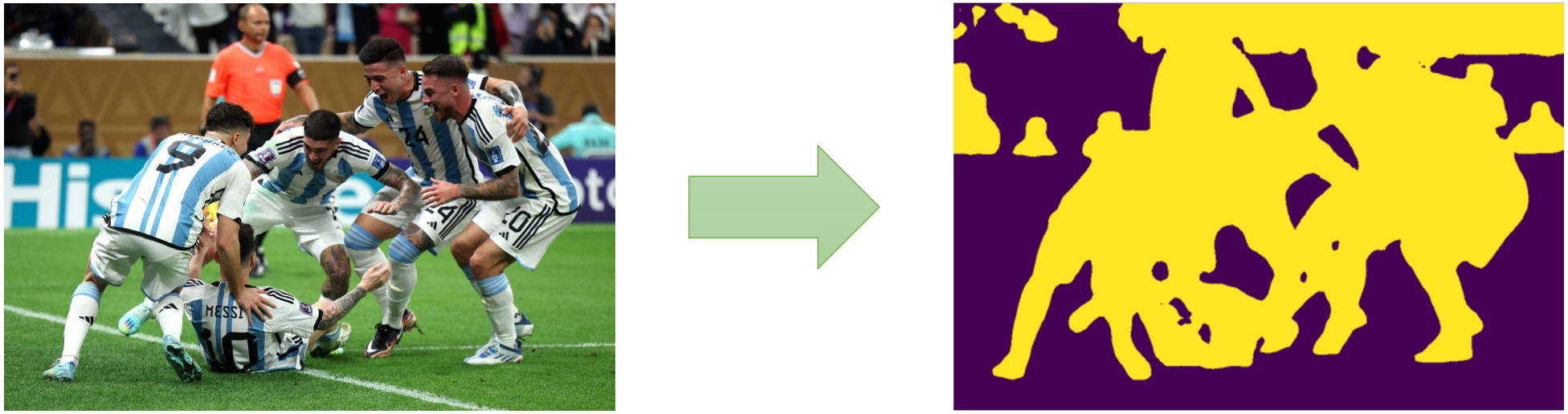


AI VIET NAM
@aivietnam.edu.vn

Segmentation

❖ Semantic segmentation

Semantic segmentation involves assigning labels to each pixel of an image in order to classify the objects within the image.

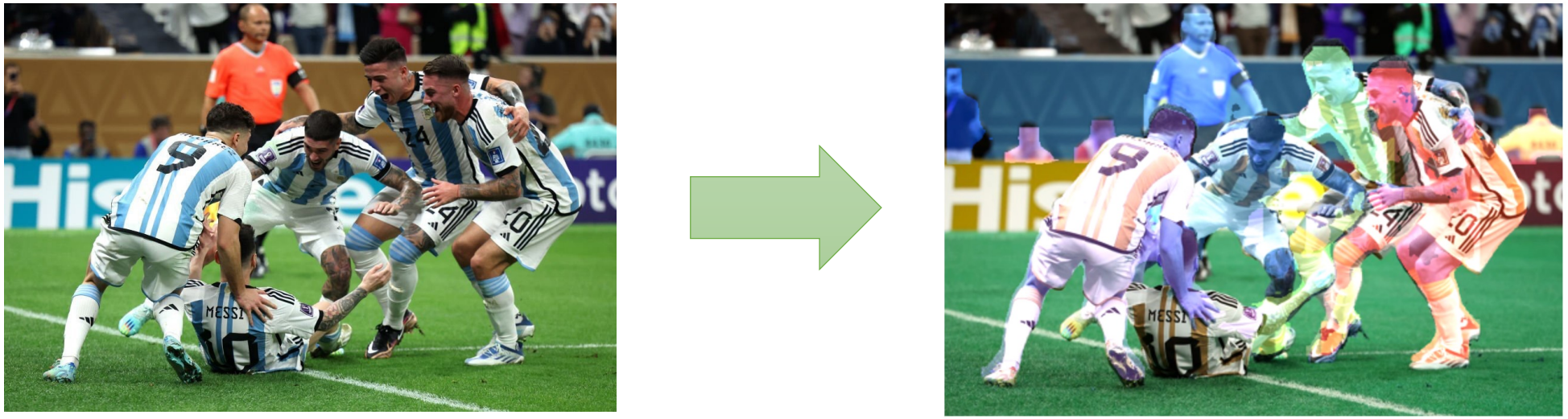


The limitation of semantic segmentation is that it cannot distinguish between objects with the same color, which poses a challenge in identifying similar objects

Segmentation

❖ Instance segmentation

Instance segmentation addresses this issue by reassigning labels to each individual object in the image.

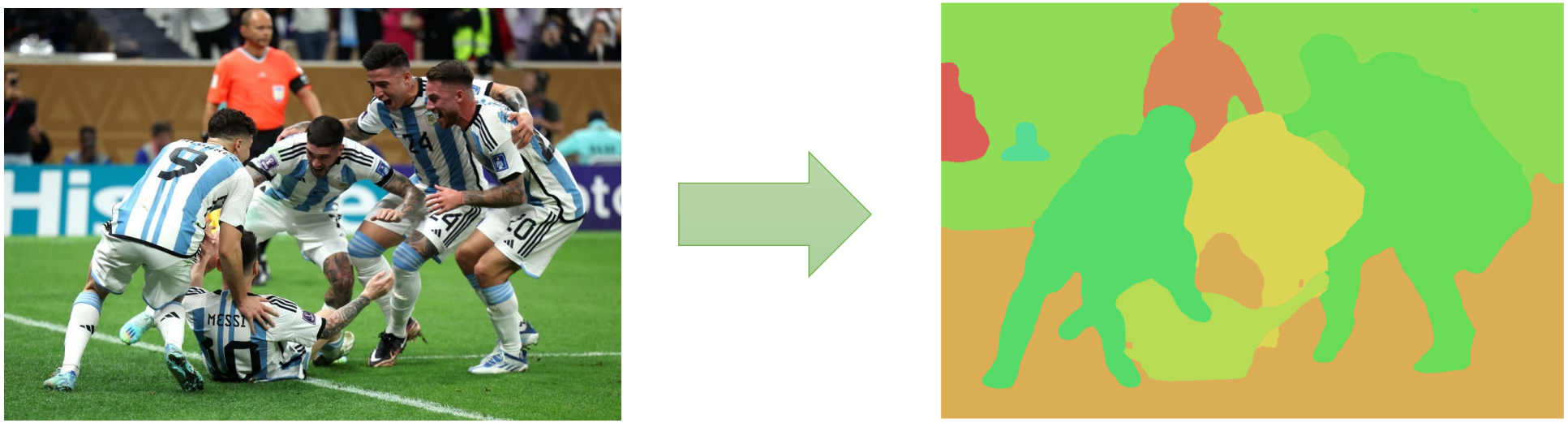


However, not all objects can be separated into distinct instances, raising the question of whether it's possible to combine both methods.

Segmentation

❖ Panoptic segmentation

Panoptic segmentation addresses this issue by combining both methods.



Semantic segmentation is used for distinct objects, while instance segmentation is applied to objects that cannot be differentiated

LIME

❖ Interception V3

Step 0: Obtain the superpixels of the image.

Obtain 4 superpixels using the
Clustering-Based Segmentation tool.

LIME

❖ Interception V3

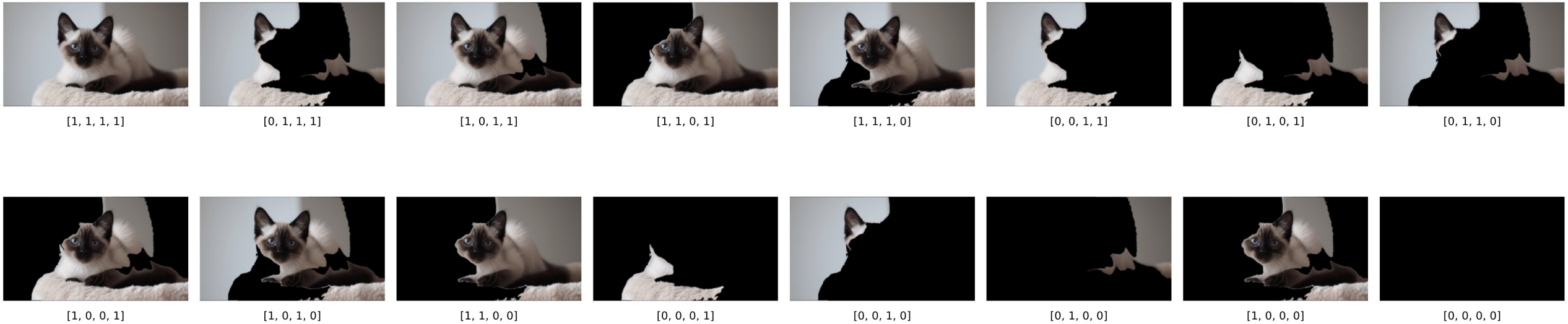
Step 1: Take sample points around X_i.

Each sample point around X_i is a noisy image created by changing the superpixels.
For example: keep superpixels 1, 2, and 4 unchanged, and modify superpixel 3.
Notation: [1, 1, 0, 1]

LIME

❖ Interception V3

Step 1: Take sample points around X_i.

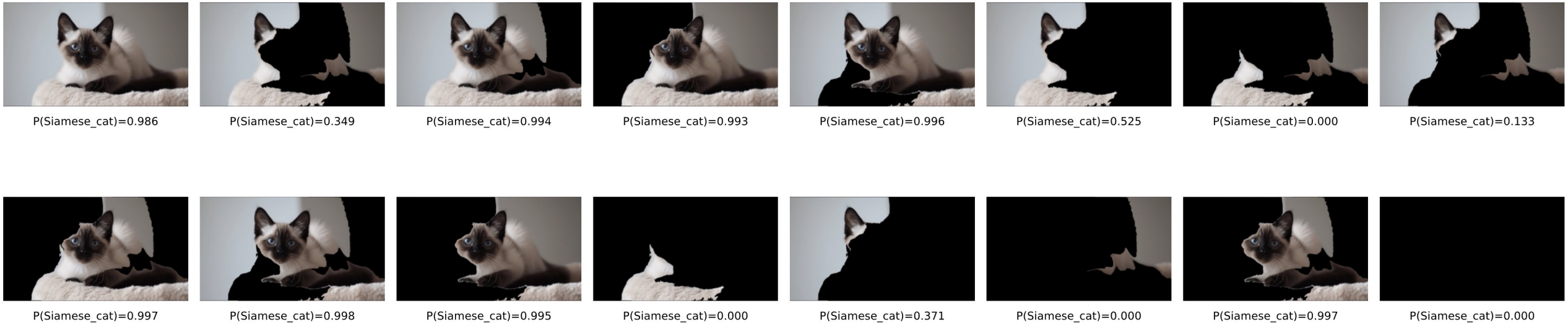


There are a total of $2^4 = 16$ permutations for the input.

LIME

❖ Interception V3

Step 2: Use a complex model to predict the label for each sample.



The prediction results of Interception V3 on all samples.

LIME

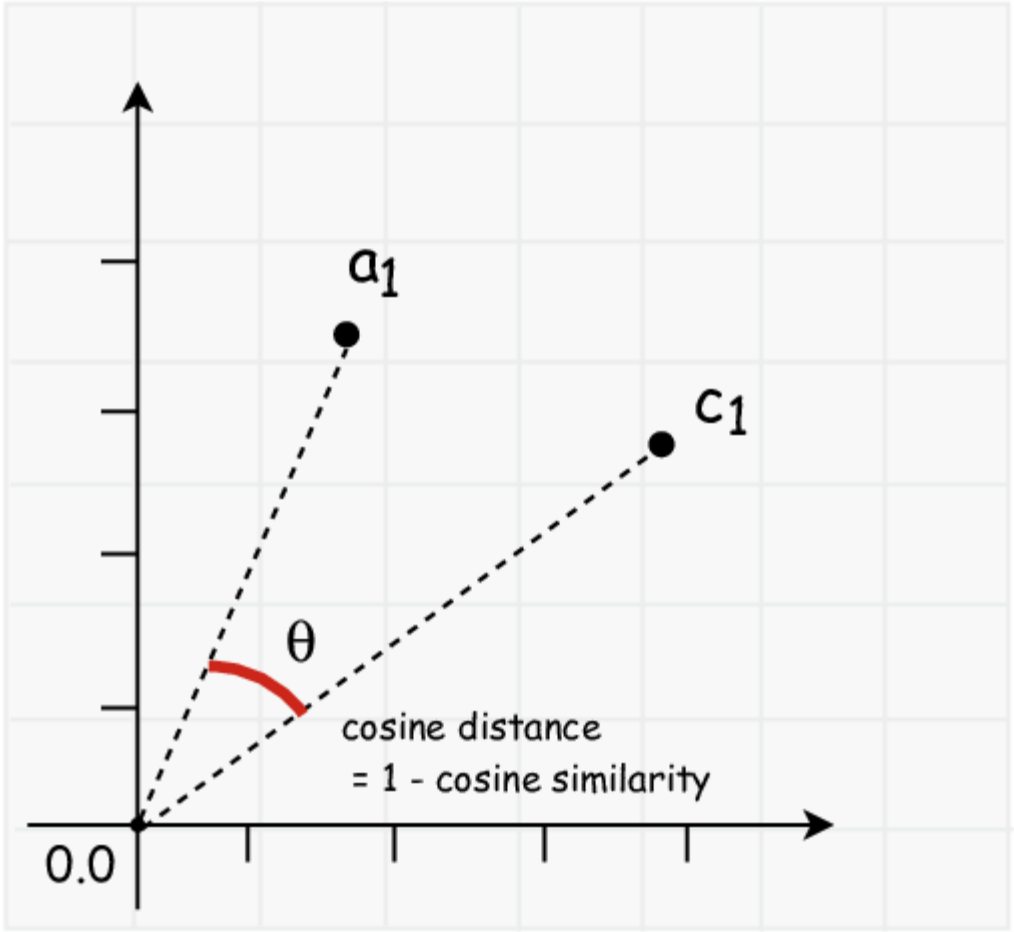
❖ Interception V3

Step 3: Assign weights to the samples based on the distance to X_i.
For two n-dimensional vectors A and B, the distance d is calculated as follows:

$$d = cosine\_distance(A, B) = 1 - cosine\_similarity(A, B)$$
$$= 1 - \frac{A \cdot B}{||A||||B||} = 1 - \frac{\Sigma_{i=1}^{n} A_i B_i}{\sqrt{\Sigma_{i=1}^{n} A_i^2}\sqrt{\Sigma_{i=1}^{n} B_i^2}}$$

For example: A = X_i = [1, 1, 1, 1] and B = [1, 0, 1, 1]

$$d = 1 - \frac{1\times1 + 1\times0 + 1\times1 + 1\times1}{\sqrt{1^2 + 1^2 + 1^2 + 1^2}\sqrt{1^2 + 0^2 + 1^2 + 1^2}} \approx 0.87$$



LIME

❖ Interception V3

Step 3: Assign weights to the samples based on the distance to X_i.

	superpixel_0	superpixel_1	superpixel_2	superpixel_3	prediction	distance
0	0.0	0.0	0.0	0.0	0.000457	1.0
1	0.0	0.0	0.0	1.0	0.000135	0.5
2	0.0	0.0	1.0	0.0	0.371148	0.5
3	0.0	0.0	1.0	1.0	0.525476	0.292893
4	0.0	1.0	0.0	0.0	0.00034	0.5

The distance has been calculated.

LIME

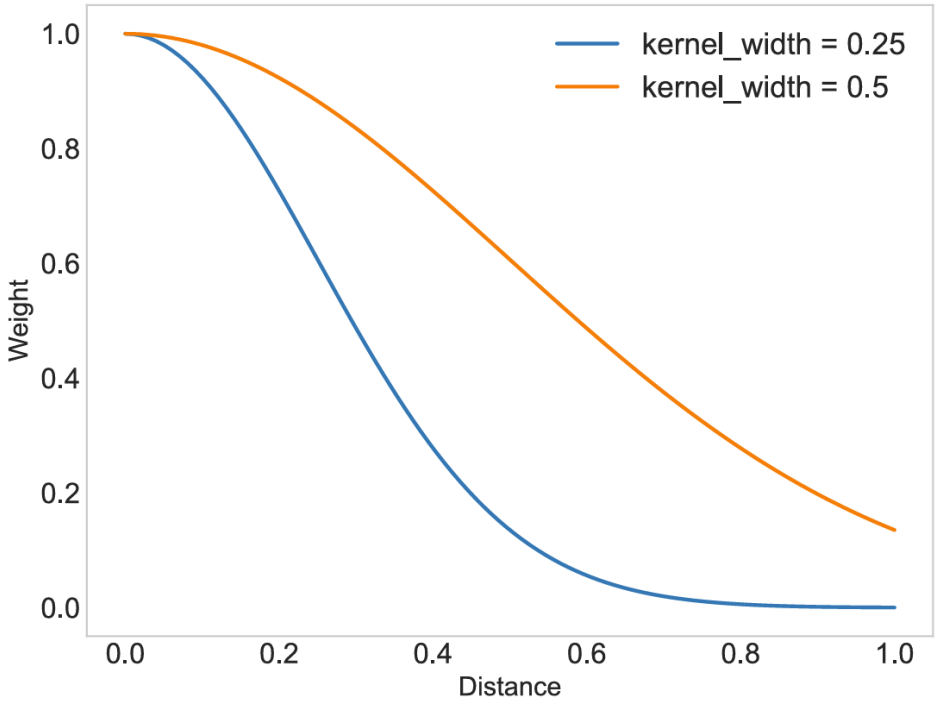
❖ Interception V3

Step 3: Assign weights to the samples based on the distance to X_i.
To convert the distance into a weight, we use the exponential kernel function:

$$w_i = exp\left(-\frac{d_i^2}{2\sigma^2}\right)$$

Large kernal size:
Expand "local" area, genetic explaination

Small kernal size:
More precise explaination



The effect of kernel width σ on the weight.

LIME

❖ Interception V3

Step 3: Assign weights to the samples based on the distance to X_i.

	superpixel_0	superpixel_1	superpixel_2	superpixel_3	prediction	weights
0	0.0	0.0	0.0	0.0	0.000457	0.0
1	0.0	0.0	0.0	1.0	0.000135	4e-06
2	0.0	0.0	1.0	0.0	0.371148	4e-06
3	0.0	0.0	1.0	1.0	0.525476	0.013714
4	0.0	1.0	0.0	0.0	0.00034	4e-06

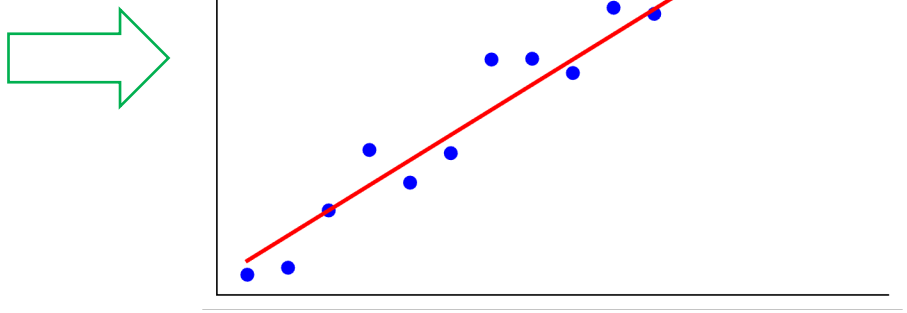
Kernel width $\sigma = 0.25$

LIME

❖ Interception V3

Step 4: Train a new simple model on the weighted samples.

	superpixel_0	superpixel_1	superpixel_2	superpixel_3	prediction	weights
0	0.0	0.0	0.0	0.0	0.000457	0.0
1	0.0	0.0	0.0	1.0	0.000135	4e-06
2	0.0	0.0	1.0	0.0	0.371148	4e-06
3	0.0	0.0	1.0	1.0	0.525476	0.013714
4	0.0	1.0	0.0	0.0	0.00034	4e-06
5	0.0	1.0	0.0	1.0	0.000264	0.013714
6	0.0	1.0	1.0	0.0	0.133267	0.013714
7	0.0	1.0	1.0	1.0	0.348899	0.407602
8	1.0	0.0	0.0	0.0	0.997444	4e-06
9	1.0	0.0	0.0	1.0	0.997458	0.013714
10	1.0	0.0	1.0	0.0	0.998116	0.013714
11	1.0	0.0	1.0	1.0	0.994312	0.407602
12	1.0	1.0	0.0	0.0	0.99474	0.013714
13	1.0	1.0	0.0	1.0	0.993305	0.407602
14	1.0	1.0	1.0	0.0	0.995779	0.407602
15	1.0	1.0	1.0	1.0	0.98623	1.0



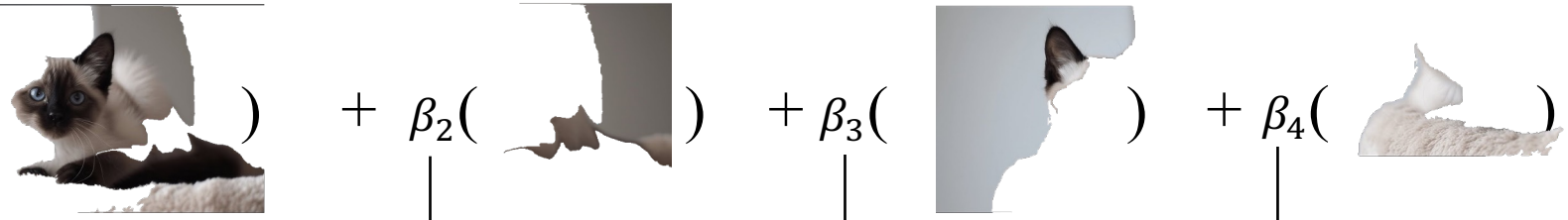
Use the transformations, Interception V3 predictions, and weights to train the **Linear Regression model**.

LIME

❖ Interception V3

Step 5: Use the simple model to explain.

$$\hat{y} = \beta_1(\ \ ) + \beta_2(\ \ ) + \beta_3(\ \ ) + \beta_4(\ \ ) + \text{bias}$$



0.878 0.011 0.147 0.08 -0.116

model.coef_ model.intercept_

By extracting the coefficients of the Linear Regression, determining the importance level of each superpixel for the model's prediction for the **Siamese** species becomes easier.

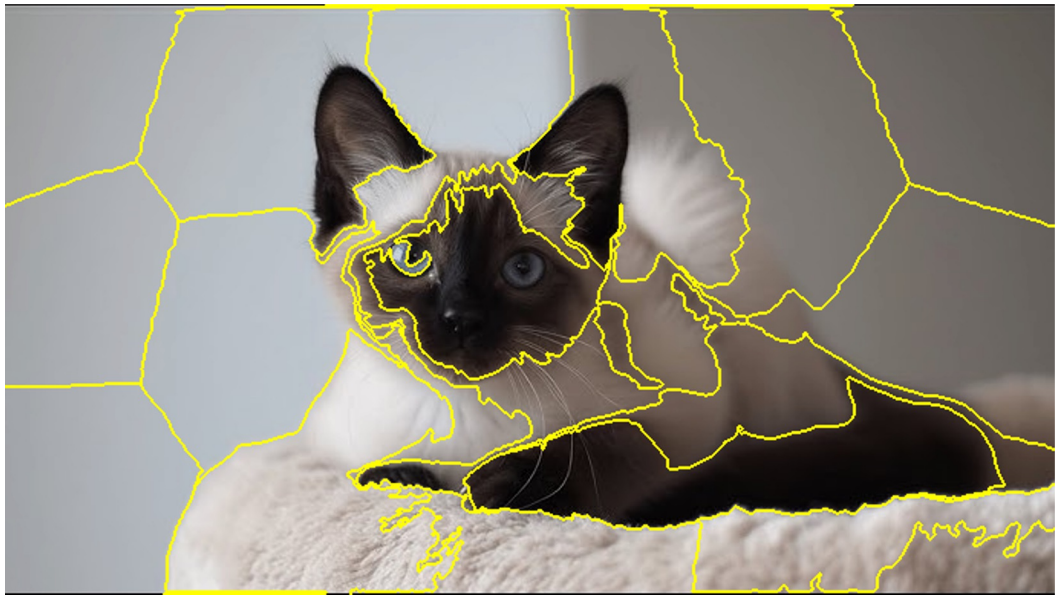
LIME

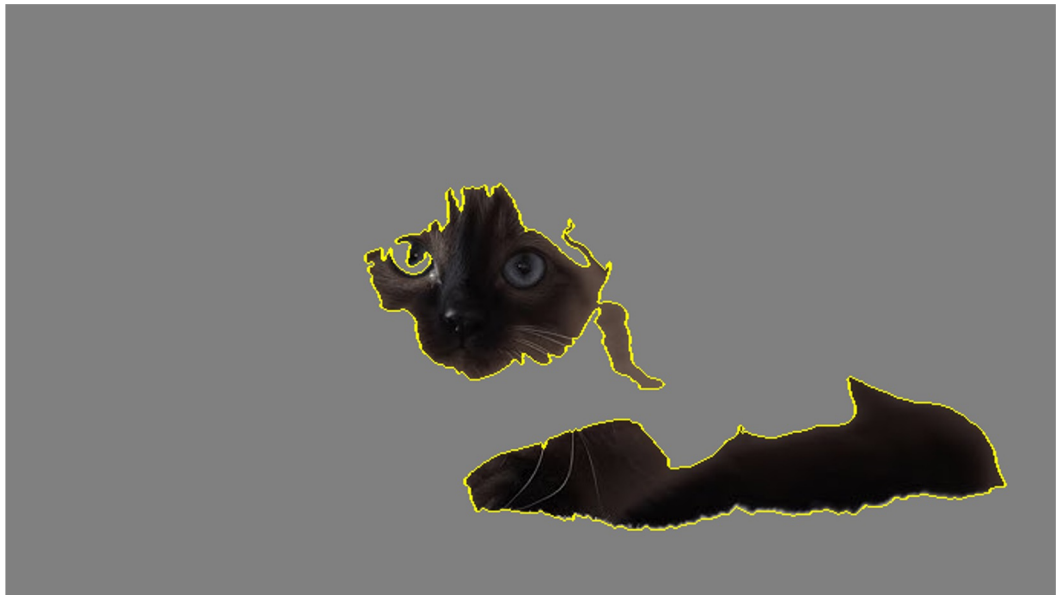
❖ Interception V3

Step 5: Use the simple model to explain.

Superpixel 1 has the highest regression coefficient, meaning it has the greatest impact on the model's decision.

LIME





The more superpixel create from the image the more specific the explanation become.

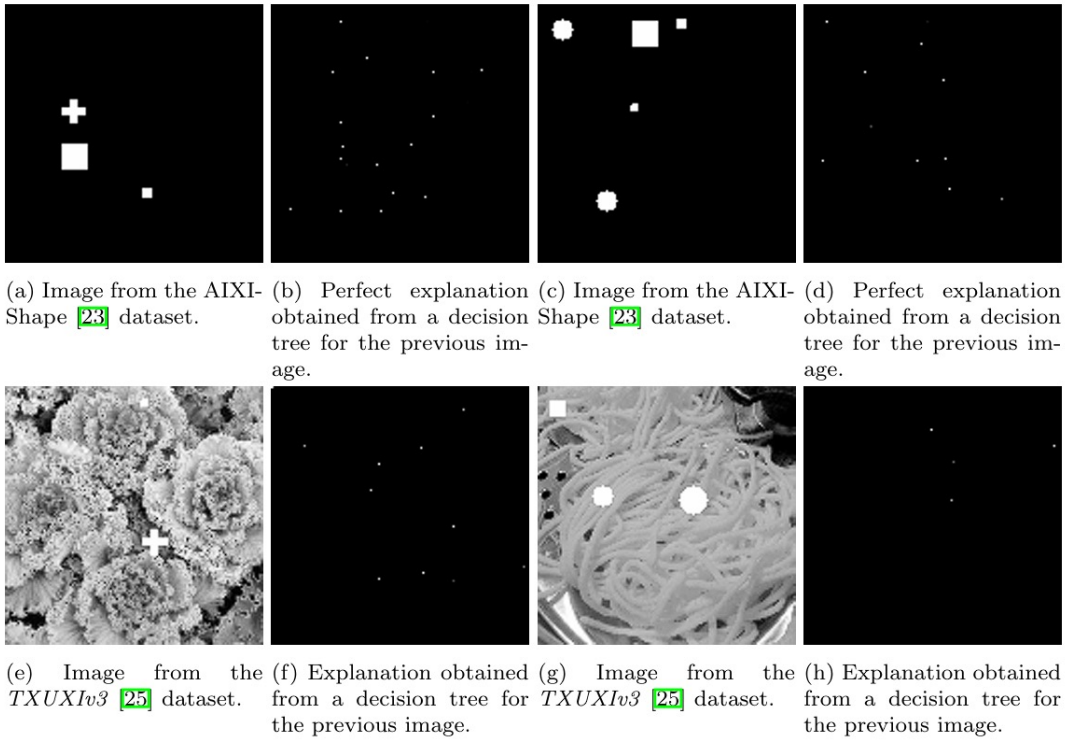
LIME - Fidelity

Focus on single sample

+

Using interpretable model

Explanation align with model



(a) Image from the AIXI-Shape [23] dataset.

(b) Perfect explanation obtained from a decision tree for the previous image.

(c) Image from the AIXI-Shape [23] dataset.

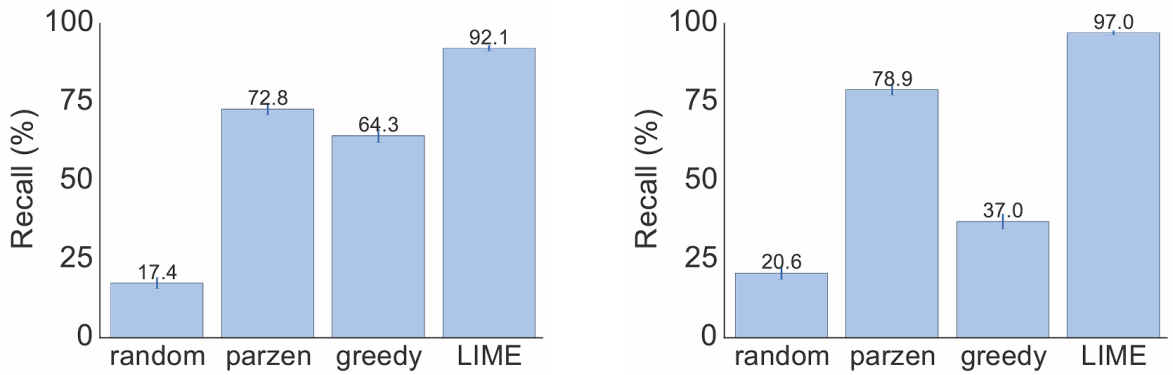
(d) Perfect explanation obtained from a decision tree for the previous image.

(e) Image from the *TXUXIv3* [25] dataset.

(f) Explanation obtained from a decision tree for the previous image.

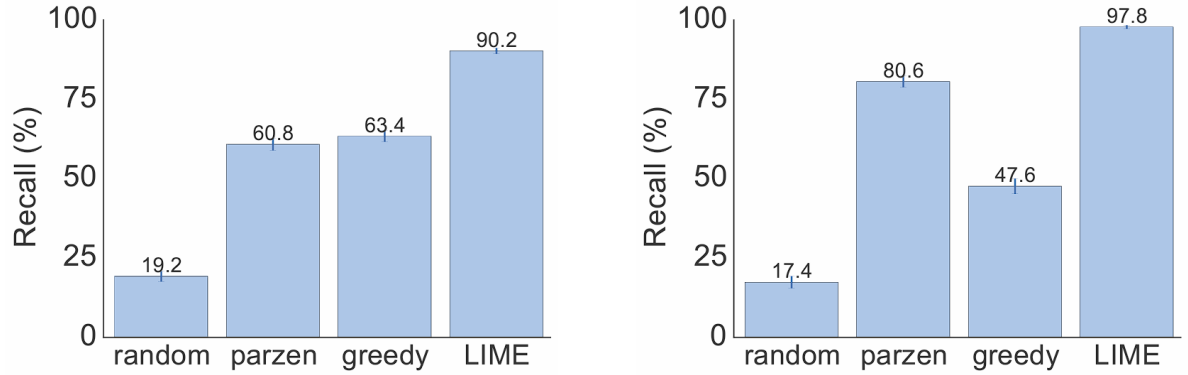
(g) Image from the *TXUXIv3* [25] dataset.

(h) Explanation obtained from a decision tree for the previous image.



(a) Sparse LR (b) Decision Tree

Figure 6: Recall on truly important features for two interpretable classifiers on the books dataset.



(a) Sparse LR (b) Decision Tree

Figure 7: Recall on truly important features for two interpretable classifiers on the DVDs dataset.

LIME

❖ Advantage

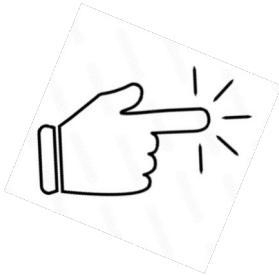
- Popular.
- Easy to understand.
- Easy to implement.

❖ Disadvantage

- Concerns about instability.
- Computationally expensive.
- Unstable explanations.
- Counterfactual examples can be unrealistic.
- Requires a large number of samples around the instance being explained.

Outline

- Why is Interpretability Need?
- Evolution of XAI
- LIME for AI Explainable
- LIME: Example



LIME

❖ Code

Load the complex model that wants to explain.

```
# Check for GPU availability and set device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Running on device: {device}")

# Initialize a pre-trained ResNet model
resnet_model = models.resnet50(pretrained=True).to(device)
resnet_model.eval()  # Set the model to evaluation mode
```

```
# Image transformation for ResNet
resnet_transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
# Load and preprocess the image
def load_and_preprocess_image(image_path):
    """Loads and preprocesses an image for ResNet."""
    img = Image.open(image_path)
    img_tensor = resnet_transform(img).unsqueeze(0).to(device)
    return img_tensor, np.array(img)

def download_image(url, save_path='downloaded_image.jpg'):
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    img.save(save_path)
    return save_path
```

LIME

❖ Code

Use the complex model to predict the class of the input image.

```python
# Predict class of input image
def predict_image(model, img_tensor):
    """Predicts the class of the input image."""
    with torch.no_grad():
        output = model(img_tensor)
    probabilities = torch.nn.functional.softmax(output[0], dim=0)
    top_prob, top_catid = torch.topk(probabilities, 5)
    return top_prob, top_catid

top_prob, top_catid = predict_image(resnet_model, img_tensor)

# Print top 5 predictions
for i in range(top_prob.size(0)):
    print(f"{top_prob[i].item():.3f} -> class index: {top_catid[i].item()} -> class: {labels[top_catid[i].item()]}")

print(f"The predicted class is: {labels[top_catid[0].item()]}")
```

```
0.995 -> class index: 32 -> class: tailed frog
0.002 -> class index: 26 -> class: common newt
0.001 -> class index: 30 -> class: bullfrog
0.001 -> class index: 31 -> class: tree frog
0.000 -> class index: 38 -> class: banded gecko
The predicted class is: tailed frog
```

LIME

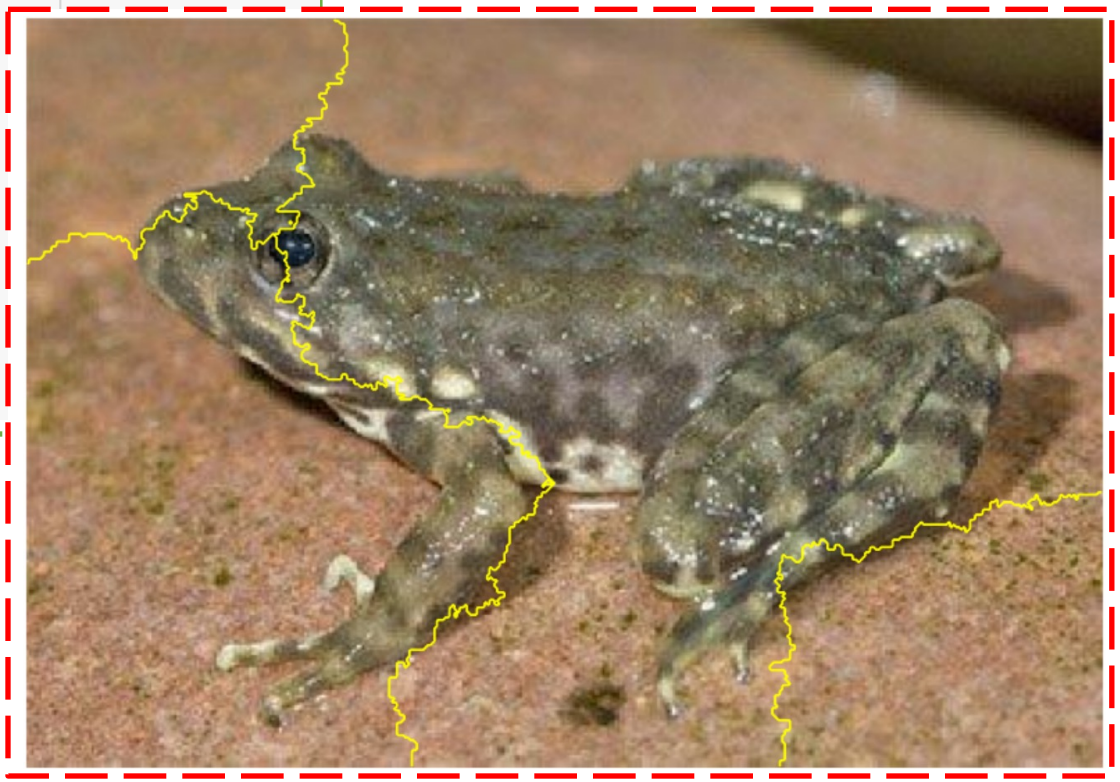
❖ Code

Create superpixels for the input image.

```python
# Generate superpixels using quickshift algorithm
def generate_superpixels(image):
    """Generates superpixels for the given image."""
    superpixels = skimage.segmentation.quickshift(image, kernel_size=21, max_dist=200, ratio=0.2)
    return superpixels

superpixels = generate_superpixels(Xi)
num_superpixels = np.unique(superpixels).shape[0]
# Visualize all superpixels with yellow border
fig, ax = plt.subplots(figsize=(10, 10))
ax.imshow(Xi)
for superpixel in np.unique(superpixels):
    mask = superpixels == superpixel
    ax.contour(mask, colors='yellow', linewidths=1)

ax.axis('off')
plt.show()
```



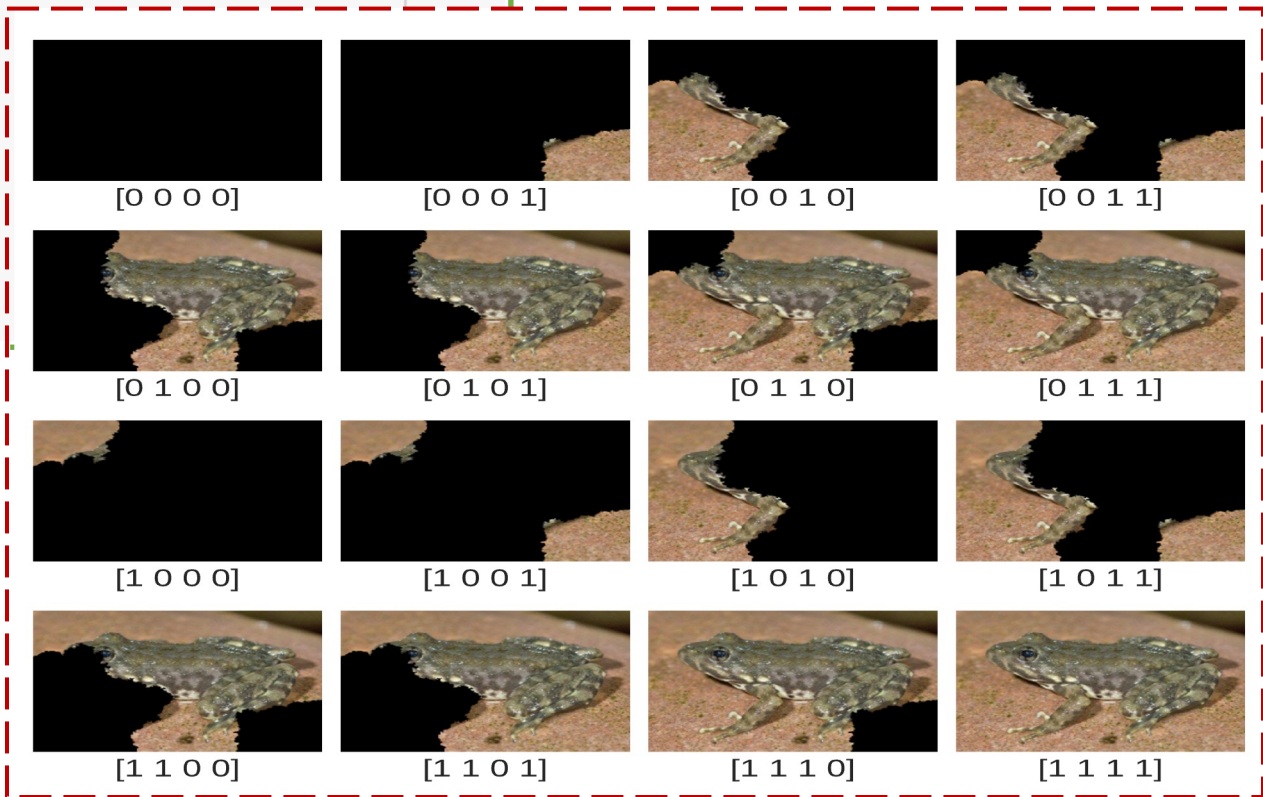
LIME

❖ Code

Create permutations for the input image.

```
# Function to perturb the image based on superpixels
def perturb_image(img, perturbation, segments):
    """Perturbs the image based on the given perturbation and superpixel segments."""
    active_pixels = np.where(perturbation == 1)[0]
    mask = np.zeros(segments.shape)
    for active in active_pixels:
        mask[segments == active] = 1
    # Expand the mask to 3 channels to keep the RGB color
    mask = np.repeat(mask[:, :, np.newaxis], 3, axis=2)
    perturbed_image = img * mask

    return perturbed_image.astype(np.uint8)
```



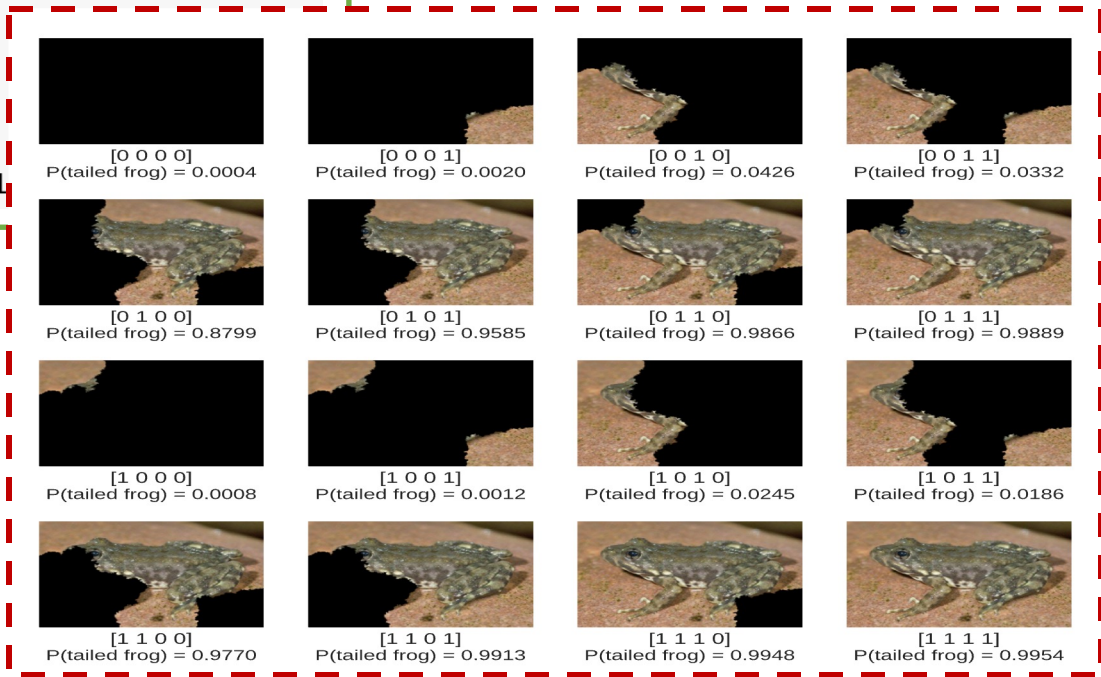
LIME

❖ Code

Use the complex model to predict the probability of the permutations.

```
# Predict classes for perturbed images
def predict_perturbed_images(perturbations, image, superpixels, model, transform):
    """Predicts the class probabilities for each perturbed image."""
    predictions = []
    for pert in perturbations:
        perturbed_img = perturb_image(image, pert, superpixels)
        perturbed_img_pil = Image.fromarray(perturbed_img.astype('uint8'))
        perturbed_img_tensor = transform(perturbed_img_pil).unsqueeze(0).to(device)
        with torch.no_grad():
            pred = model(perturbed_img_tensor)
        predictions.append(pred.cpu().numpy())
    return np.array(predictions)

predictions = predict_perturbed_images(perturbations, Xi, superpixels, resnet_model
```



AI VIET NAM
@aivietnam.edu.vn

LIME

❖ Code

Assign weight to each permutation based on the exponential kernel algorithm.

```python
# Compute distances and weights
def compute_distances_and_weights(perturbations, num_superpixels, kernel_width=0.25):
    """Computes distances between perturbations and the original image, and calculates weights."""
    original_image = np.ones(num_superpixels)[np.newaxis, :]
    distances = sklearn.metrics.pairwise_distances(perturbations, original_image, metric='cosine').ravel()
    weights = np.sqrt(np.exp(-(distances**2) / kernel_width**2))
    return distances, weights

distances, weights = compute_distances_and_weights(perturbations, num_superpixels)
```

$$w_i = exp\left(-\frac{d_i^2}{2\sigma^2}\right)$$

$$\begin{aligned} d &= cosine\_distance(A, B) \\ &= 1 - cosine\_similarity(A, B) \\ &= 1 - \frac{A \cdot B}{||A||||B||} = 1 - \frac{\Sigma_{i=1}^{n} A_i B_i}{\sqrt{\Sigma_{i=1}^{n} A_i^2}\sqrt{\Sigma_{i=1}^{n} B_i^2}} \end{aligned}$$

	Superpixel 0	Superpixel 1	Superpixel 2	Superpixel 3	d	w	y
0	0.0	0.0	0.0	0.0	1.000000	0.000335	0.000444
1	0.0	0.0	0.0	1.0	0.500000	0.135335	0.001989
2	0.0	0.0	1.0	0.0	0.500000	0.135335	0.042633
3	0.0	0.0	1.0	1.0	0.292893	0.503440	0.033225
4	0.0	1.0	0.0	0.0	0.500000	0.135335	0.879854
5	0.0	1.0	0.0	1.0	0.292893	0.503440	0.958524
6	0.0	1.0	1.0	0.0	0.292893	0.503440	0.986611
7	0.0	1.0	1.0	1.0	0.133975	0.866240	0.988864
8	1.0	0.0	0.0	0.0	0.500000	0.135335	0.000794
9	1.0	0.0	0.0	1.0	0.292893	0.503440	0.001239
10	1.0	0.0	1.0	0.0	0.292893	0.503440	0.024485
11	1.0	0.0	1.0	1.0	0.133975	0.866240	0.018585
12	1.0	1.0	0.0	0.0	0.292893	0.503440	0.977022
13	1.0	1.0	0.0	1.0	0.133975	0.866240	0.991281
14	1.0	1.0	1.0	0.0	0.133975	0.866240	0.994822
15	1.0	1.0	1.0	1.0	0.000000	1.000000	0.995449

LIME

❖ Code

Train a new simple model on the weighted samples and get the highest regression coefficient

```
class_to_explain = top_catid[0].item()

simpler_model = LinearRegression()
simpler_model.fit(X=perturbations, y=probabilities[:, 0, class_to_explain], sample_weight=weights)

coeff = simpler_model.coef_

def get_top_features(coeff, num_top_features=2):
    """Identifies the top features based on the linear model coefficients."""
    top_features = np.argsort(coeff)[-num_top_features:]
    return top_features

num_top_features = 1

top_features = get_top_features(coeff, num_top_features)
```

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

$$\text{Weighted SSE} = \sum_{i=1}^{n} w_i(y_i - \hat{y}_i)^2$$

LIME

❖ Code

Explain the model's decision.

```
# Generate and display the LIME explanation
def show_lime_explanation(image, top_features, superpixels, num_superpixels):
    """
    Generates and displays the LIME explanation image, overlaying the
    explanation on top of the original image.
    """
    fig, ax = plt.subplots(figsize=(7, 7))

    # Create a mask where 1 indicates pixels to highlight (top features)
    perturbation = np.zeros(num_superpixels)
    perturbation[top_features] = True

    active_pixels = np.where(perturbation == 1)[0]
    mask = np.zeros(superpixels.shape)
    for active in active_pixels:
        mask[superpixels == active] = 1
    perturbed_image = copy.deepcopy(image)
    perturbed_image = perturbed_image * mask[:, :, np.newaxis]

    # Display the image
    # plt.imshow(perturbed_image.astype(np.uint8))
    ax.imshow(perturbed_image.astype(np.uint8))
    ax.axis('off')

    plt.show()

show_lime_explanation(Xi, top_features, superpixels, num_superpixels)
```

Superpixel 2 has the greatest impact on the model's decision.

AI VIET NAM
@aivietnam.edu.vn

Text Data

🧪 Problem:

You're given this sentence:

> "The movie was absolutely wonderful and the acting was superb."

Your model predicts: **Positive Sentiment (0.95 probability)**

You want to know: **Why did the model make this prediction?**

Text Data

Step 1: Select the instance

"The movie was absolutely wonderful and the acting was superb."

Text Data

"The movie was absolutely wonderful and the acting was superb."

Step 2: Generate perturbed versions of the text

1. “The movie was absolutely and the acting was superb.”
2. “The movie was absolutely wonderful and the acting was.”
3. “movie was absolutely wonderful and acting was superb.”
4. “The movie was wonderful acting was superb.”

Text Data

"The movie was absolutely wonderful and the acting was superb."

Step 3: Get predictions for the perturbed samples

1. "The movie was absolutely and the acting was superb."
2. "The movie was absolutely wonderful and the acting was."
3. "movie was absolutely wonderful and acting was superb."
4. "The movie was wonderful acting was superb."

→ Transformer Model →

- Perturbed #1 → 0.81 (positive)
- Perturbed #2 → 0.64 (positive)
- Perturbed #3 → 0.89 (positive)
- Perturbed #4 → 0.73 (positive)

Text Data

"The movie was absolutely wonderful and the acting was superb."

Step 4: Assign similarity weights

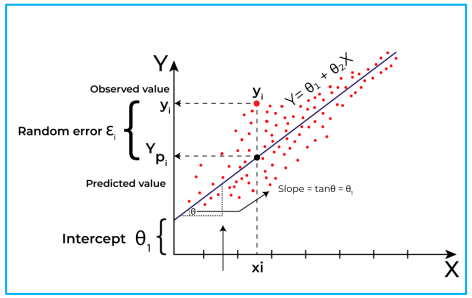
LIME gives **higher weight to samples more similar to the original sentence** (i.e., fewer words removed = more similar).

Text Data

"The movie was absolutely wonderful and the acting was superb."

Step 5: Train an interpretable model

Input =
binary features (whether each word was present)



Target =
sentiment prediction from the black-box model

Sample weight = similarity to original sentence

Text Data

"The movie was absolutely wonderful and the acting was superb."

Step 5: Show Explanation

Word	Weight
wonderful	+0.45
superb	+0.32
absolutely	+0.20
acting	+0.15
movie	+0.05

Text Data

"The movie was absolutely wonderful and the acting was superb."

```python
from lime.lime_text import LimeTextExplainer
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer

# Sample dataset
texts = ["The movie was great", "I hated the movie", "It was wonderful", "Terrible plot",
labels = [1, 0, 1, 0, 1]  # 1 = Positive, 0 = Negative

# Train a simple model
vectorizer = TfidfVectorizer()
clf = LogisticRegression()
pipeline = make_pipeline(vectorizer, clf)
pipeline.fit(texts, labels)

# Explain this instance
explainer = LimeTextExplainer(class_names=["Negative", "Positive"])
text_to_explain = "The movie was absolutely wonderful and the acting was superb."
exp = explainer.explain_instance(text_to_explain, pipeline.predict_proba, num_features=6)

# Show explanation
exp.show_in_notebook()
```