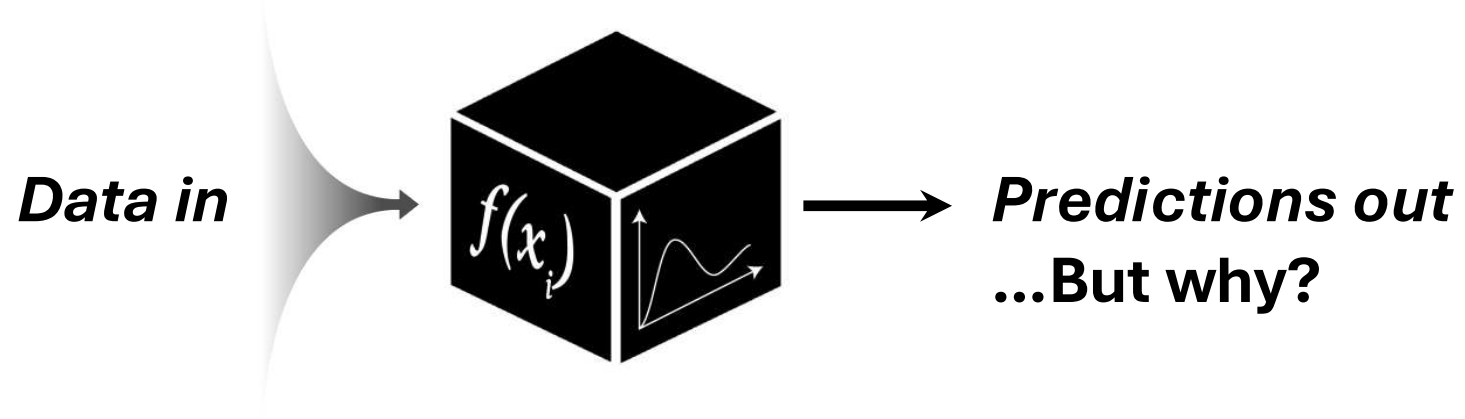


COM SCI M148, Lecture 16:
Interpreting Black-Box Models

Prof. Nathan Szymanski
nszymanski@seas.ucla.edu
Department of Materials Science

UCLA

Samueli
School of Engineering

From linear regression to neural networks

Linear regression

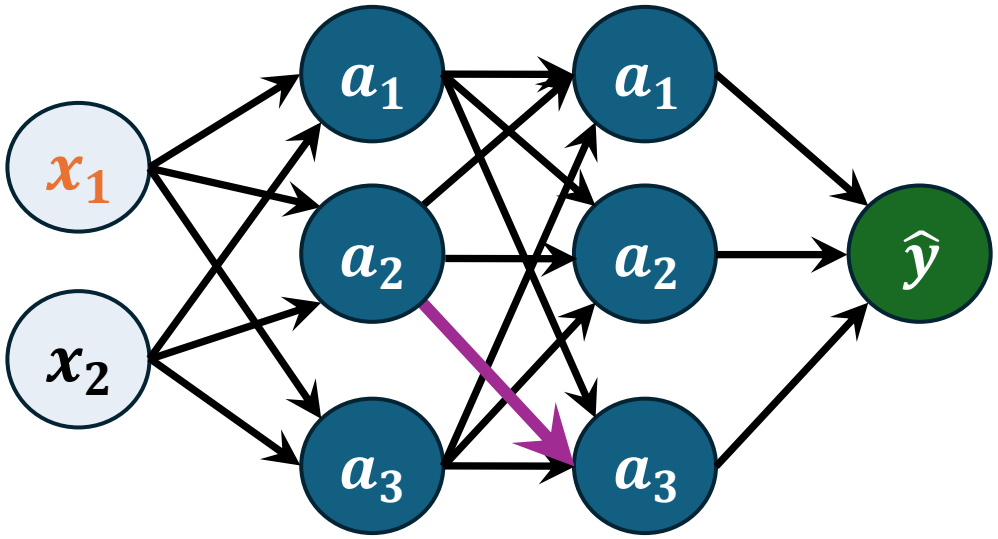
$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

These signs and magnitudes of these coefficients give us direct insight

How does each input feature affect the output?

→ **You have a clear answer!**

Neural networks

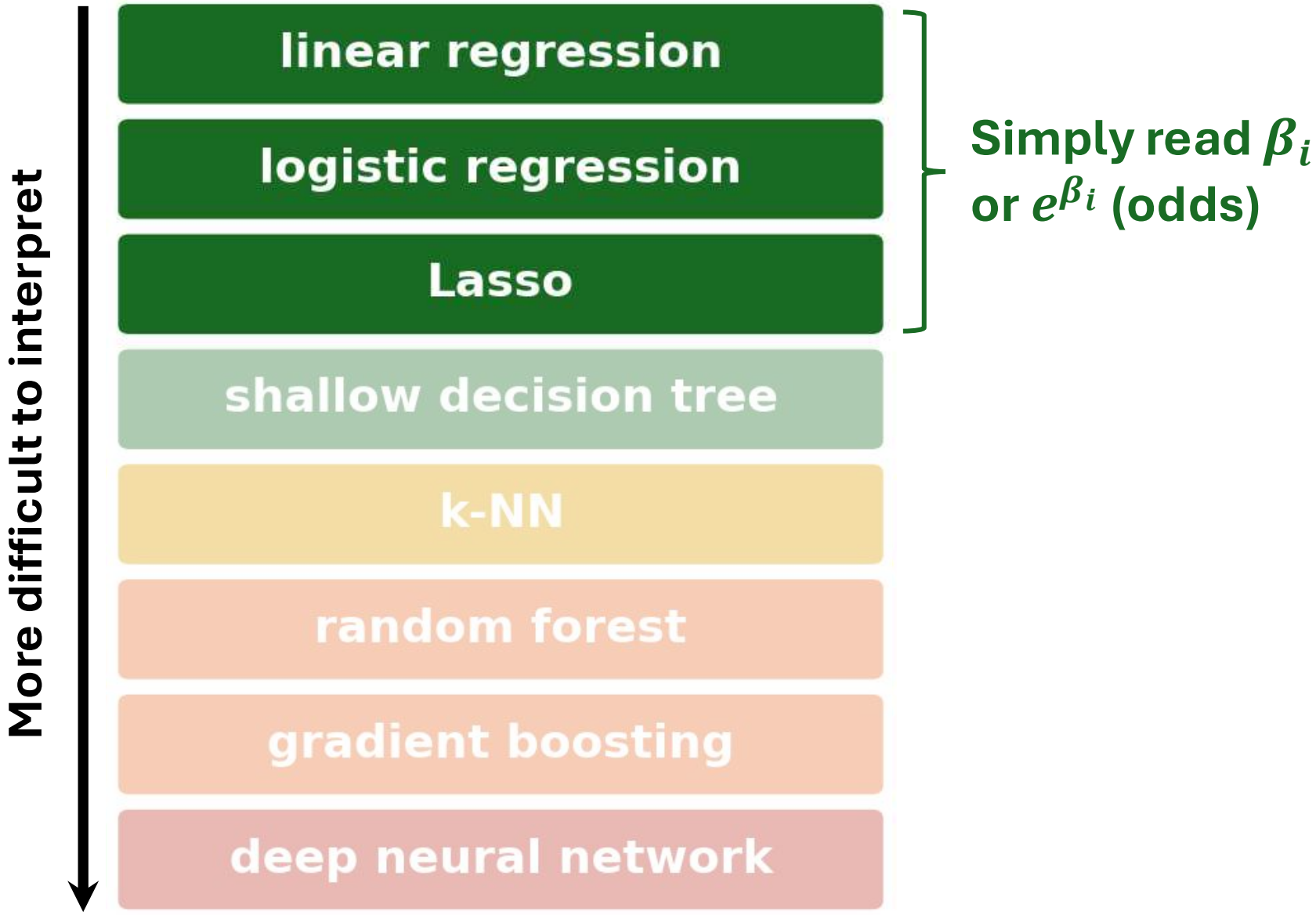


Suppose I asked you:

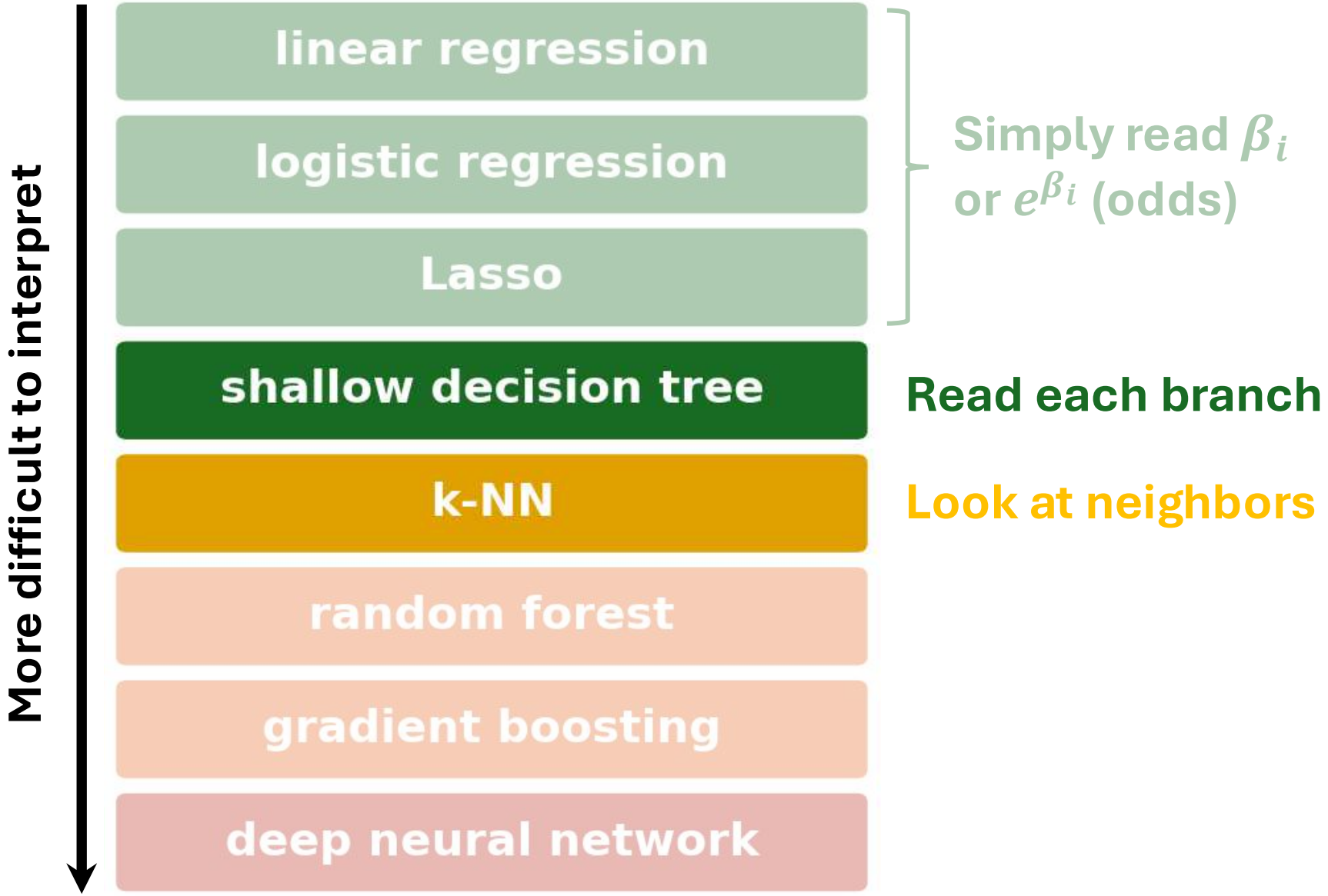
- How does x_1 relate to $\hat{y}$?
- $w_{32}^{(2)}$ is large…what does that mean?

You'd probably have no idea!

The “interpretability ladder”



The “interpretability ladder”



The “interpretability ladder”

More difficult to interpret

- linear regression
- logistic regression
- Lasso

Simply read β_i or e^{β_i} (odds)

- shallow decision tree

Read each branch

- k-NN

Look at neighbors

- random forest
- gradient boosting

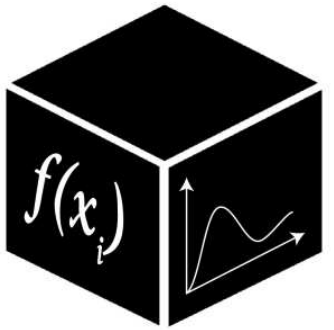
Each tree is simple but 1000s of trees …not so much

- deep neural network

Often *millions* of weights



There is often a **tradeoff** between model capacity and interpretability



“Black box”

Why do we care about interpretability?

Reason #1: We want to ***explain*** correlations and prove/disprove a hypothesis

Does the predictor i affect the response? ***Equivalent*** **Is the coefficient β_i sufficiently non-zero?**

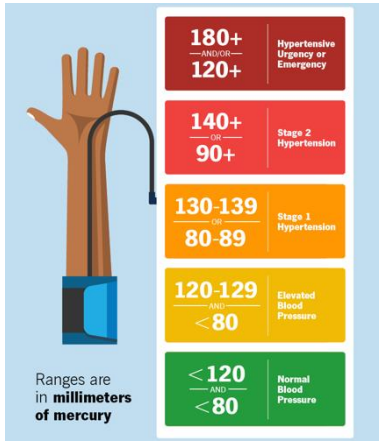
Does studying more lead to better exam scores? ***Equivalent*** Is the slope of $\beta_{\text{hrs studied}}$ large and positive?

Reason #2: To instill ***trust*** in our model – test accuracy alone can be misleading!

What the average test accuracy doesn't tell you about:

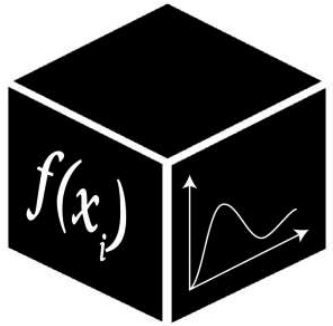
- **Data leakage**
- **Outliers with high error**
- **Changes between our test data and "real" data**

Interpretable models are easier to trust









$P(\text{operate}|x)$

Which would you trust more?

A model that says "**Yes, you should operate because the patient has high blood pressure and a lesion in the X-rays/MRI scan.**"

A model that says "**Yes, you should operate because…Idk, but trust me, my test accuracy is 95%**"

Another reason for interpretability: Debugging

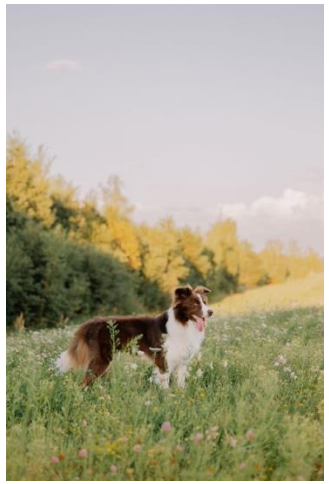
Wolves

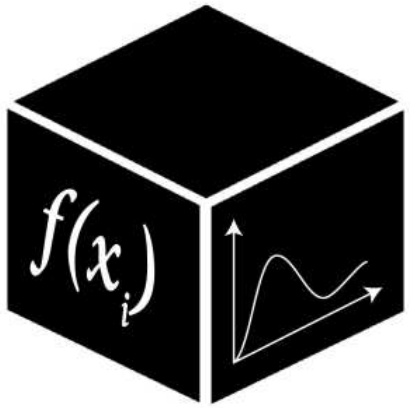





Dogs





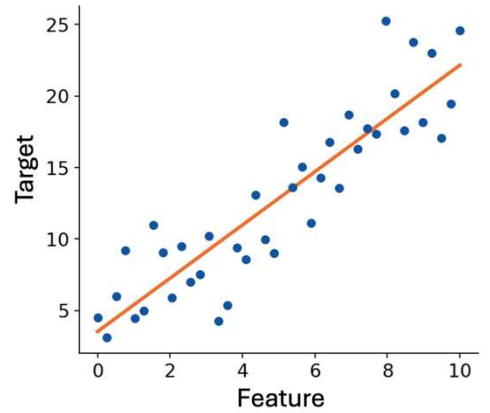


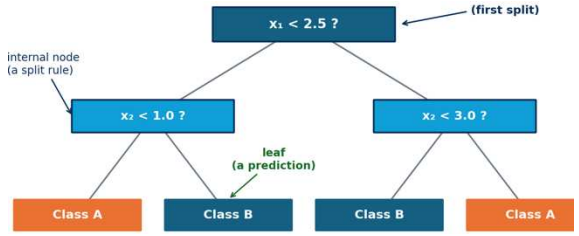
With interpretable predictions, we figure out that our model uses the background (white *vs*. green) to decide

"I used this part of the image to classify it as a wolf!"

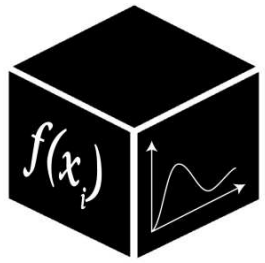
To get interpretability, we have two options

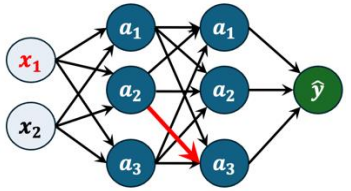
1. Use a model that is **interpretable by construction**
   - Linear and logistic regression models (β_i)
   - Shallow decision trees, where you can follow the logic branch-by-branch
   - Simpler is better, if possible, but sometimes we need very complex models to fit the data





2. Keep the black-box model and **explain it after the fact**
   - Fit a neural network or random forest, then perform additional **analysis once training is done** to get an ***explanation*** of the model's predictions

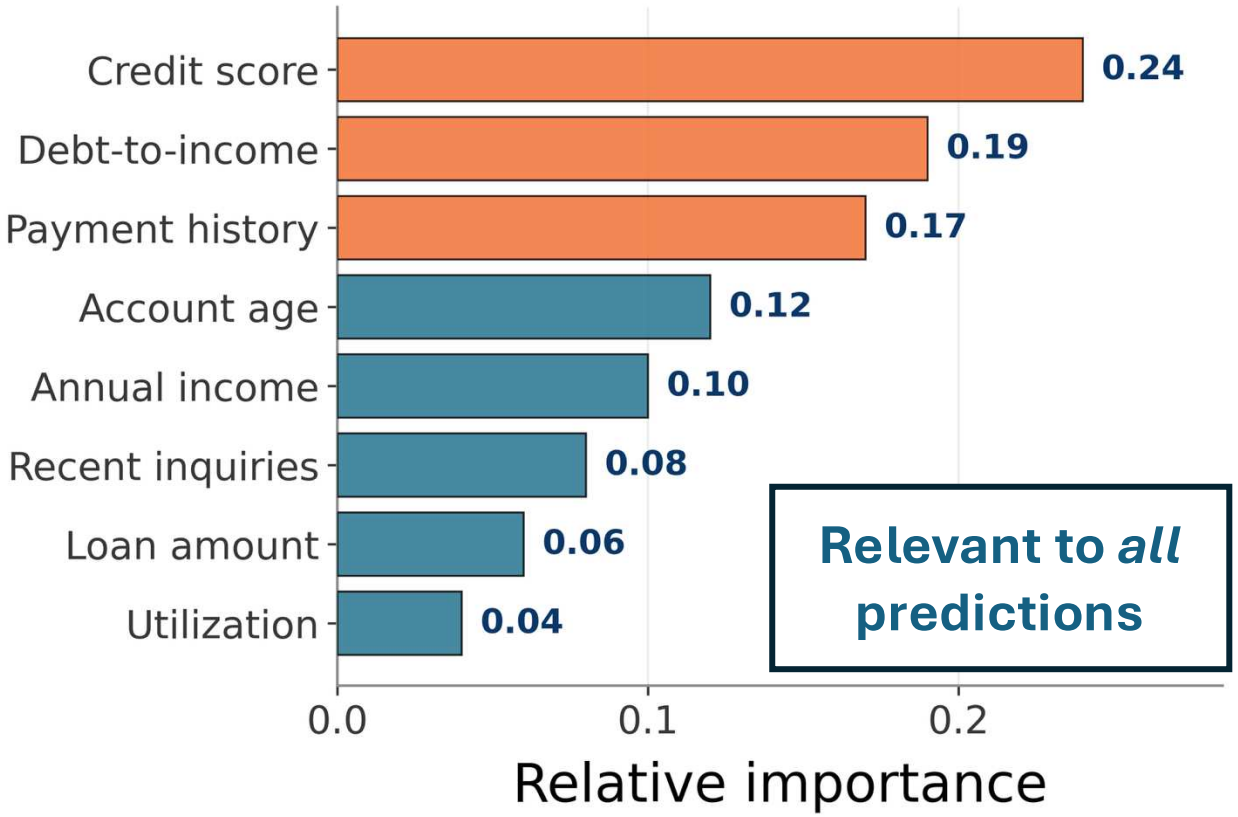




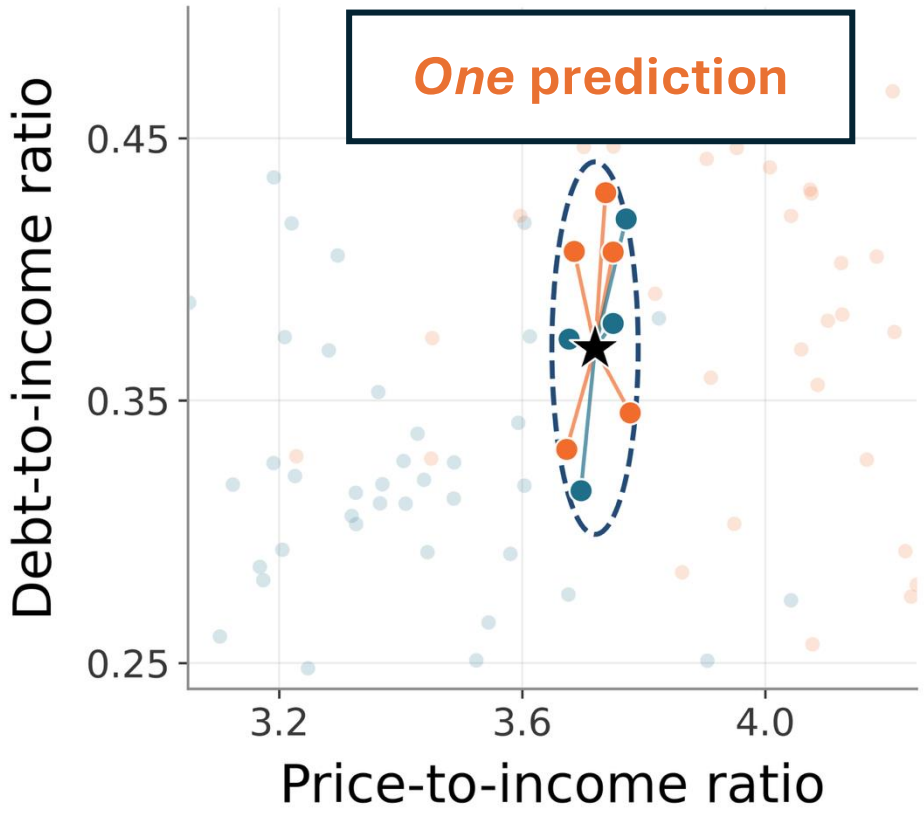
What do I mean by “explanation”

Global *vs.* local explanations, and **model-specific *vs.* model-agnostic**

Global *vs.* local explanations

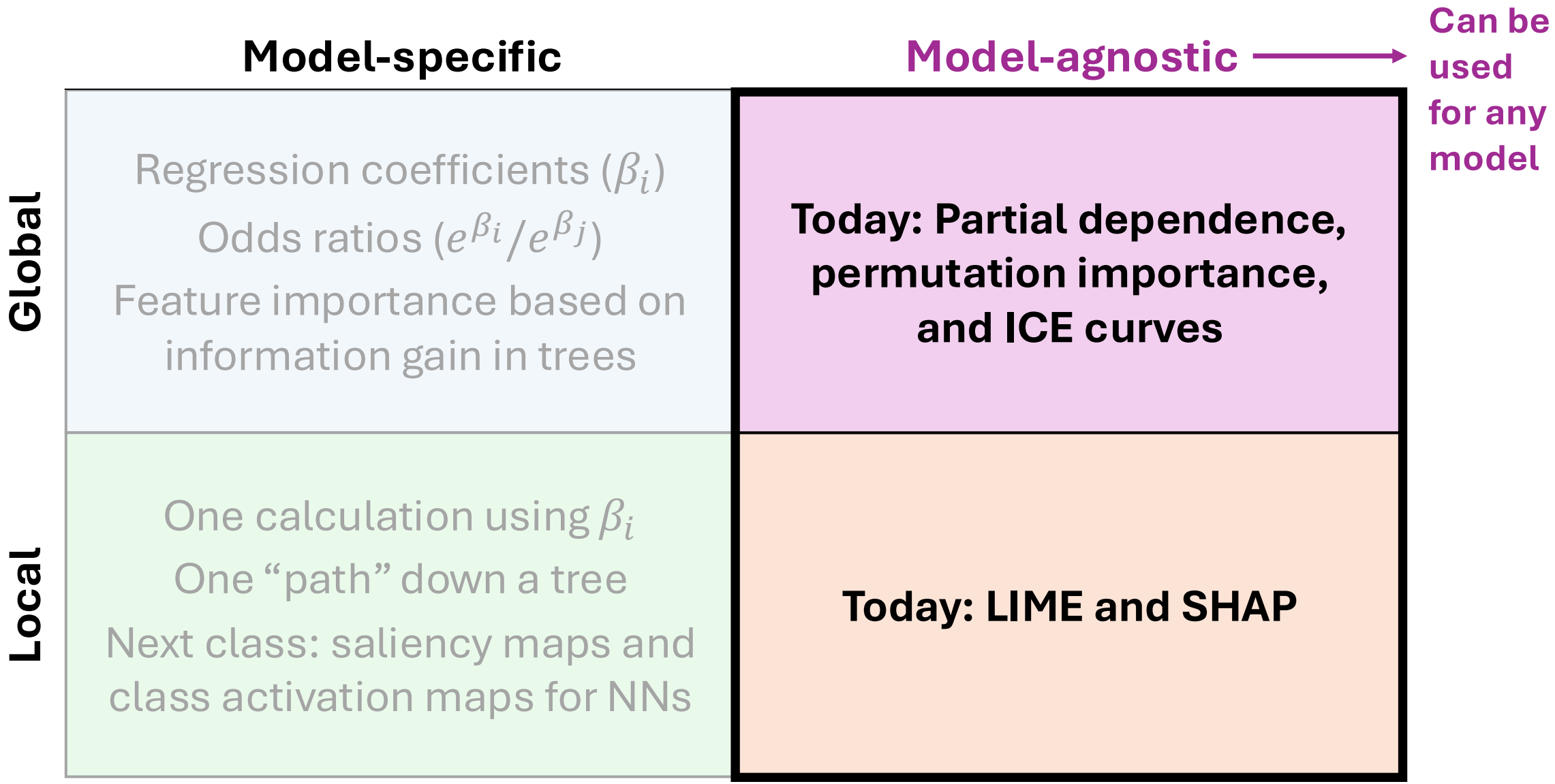


A ***global*** question might be: Which features are generally most important to this model?



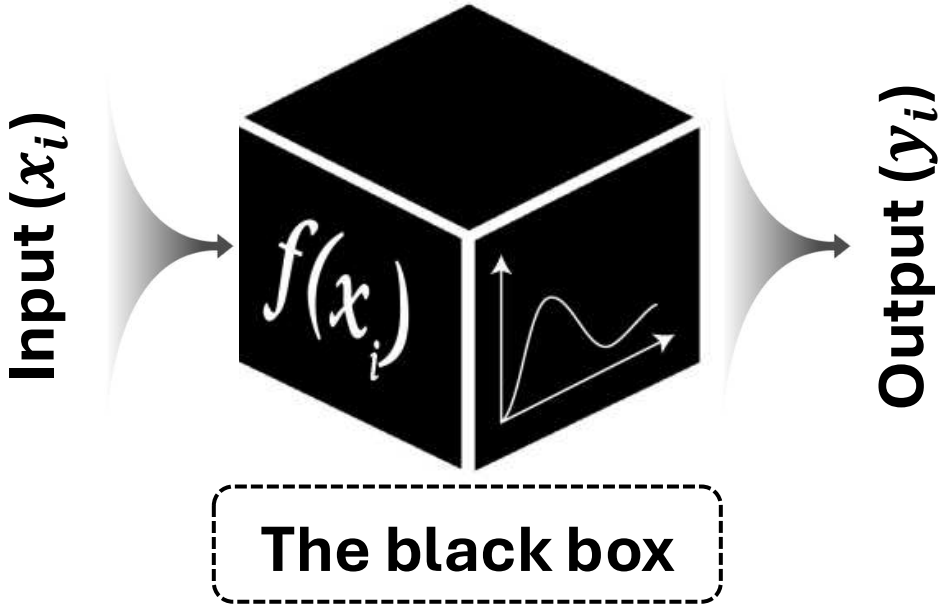
A ***local*** question might be: Why does the model make this **specific prediction?**

Model-specific *vs.* agnostic explanations



Model-specific *vs.* agnostic explanations

Let's start by focusing on **model-agnostic** explanations



The black box

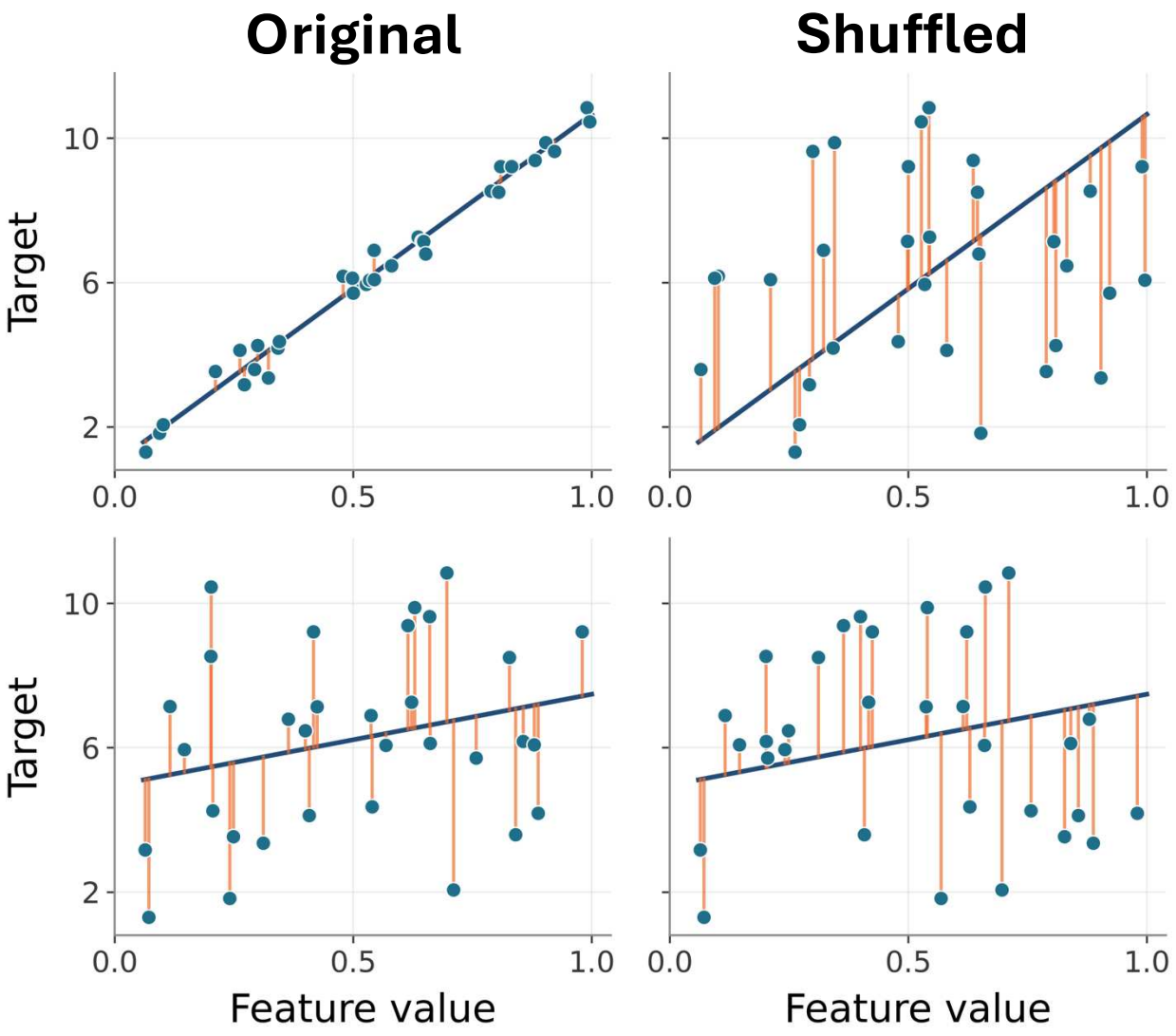
This could be anything! A regression model, decision tree, neural network, crystal ball, …



Model-agnostic

Today: Partial dependence, permutation importance, and ICE curves	**Global**
Today: LIME and SHAP	**Local**

Permutation importance: General idea



Model-agnostic

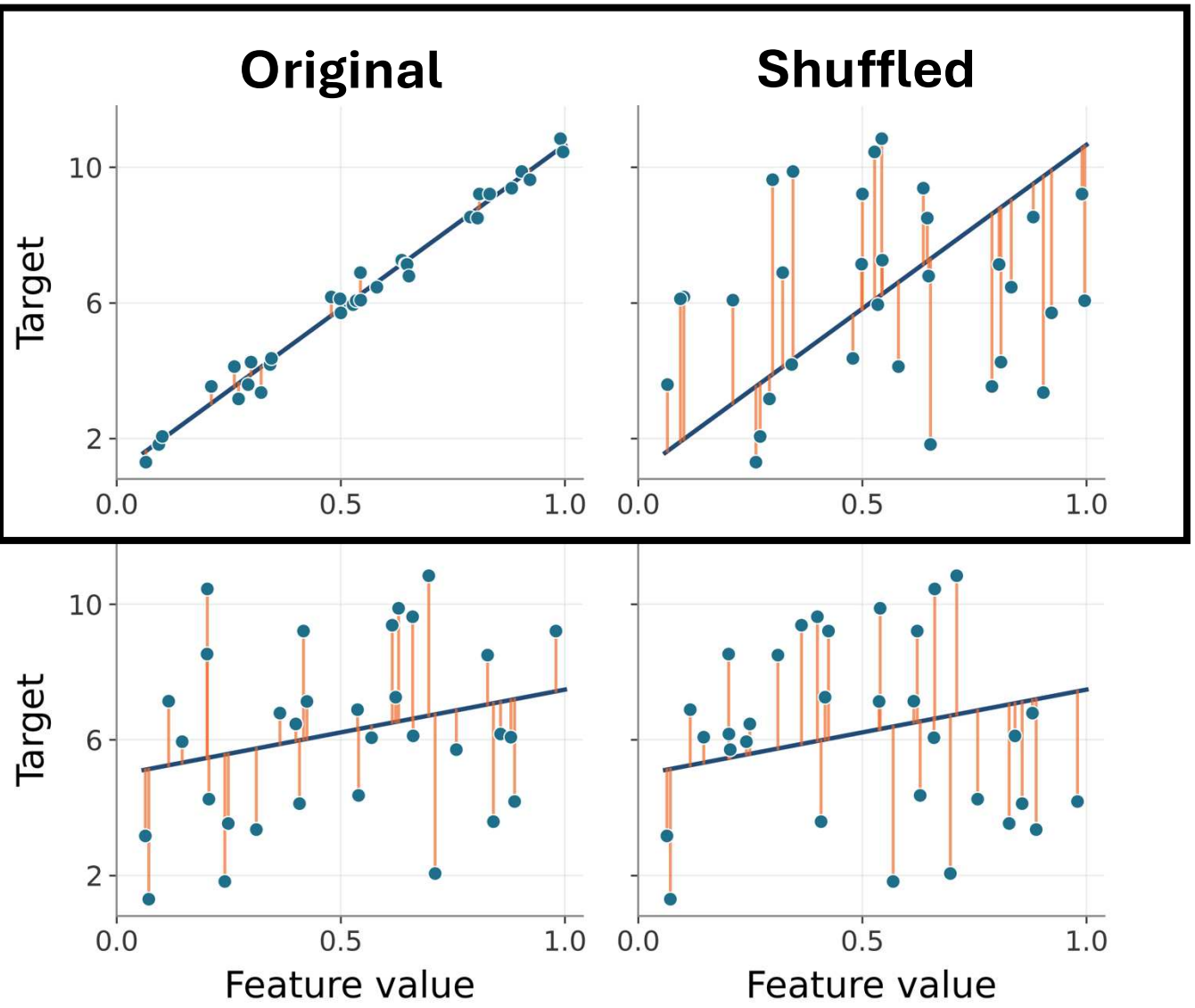
Today: Partial dependence, **permutation importance,** and ICE curves

Global

Basic idea of permutation importance:

If a feature is important, then randomly scrambling its values should make the model perform worse.

Permutation importance: General idea



Here, the feature was clearly important as it had a strong correlation with the target. Our linear model* has much higher error if we randomly swap inputs (x_i) while fixing the output (y).

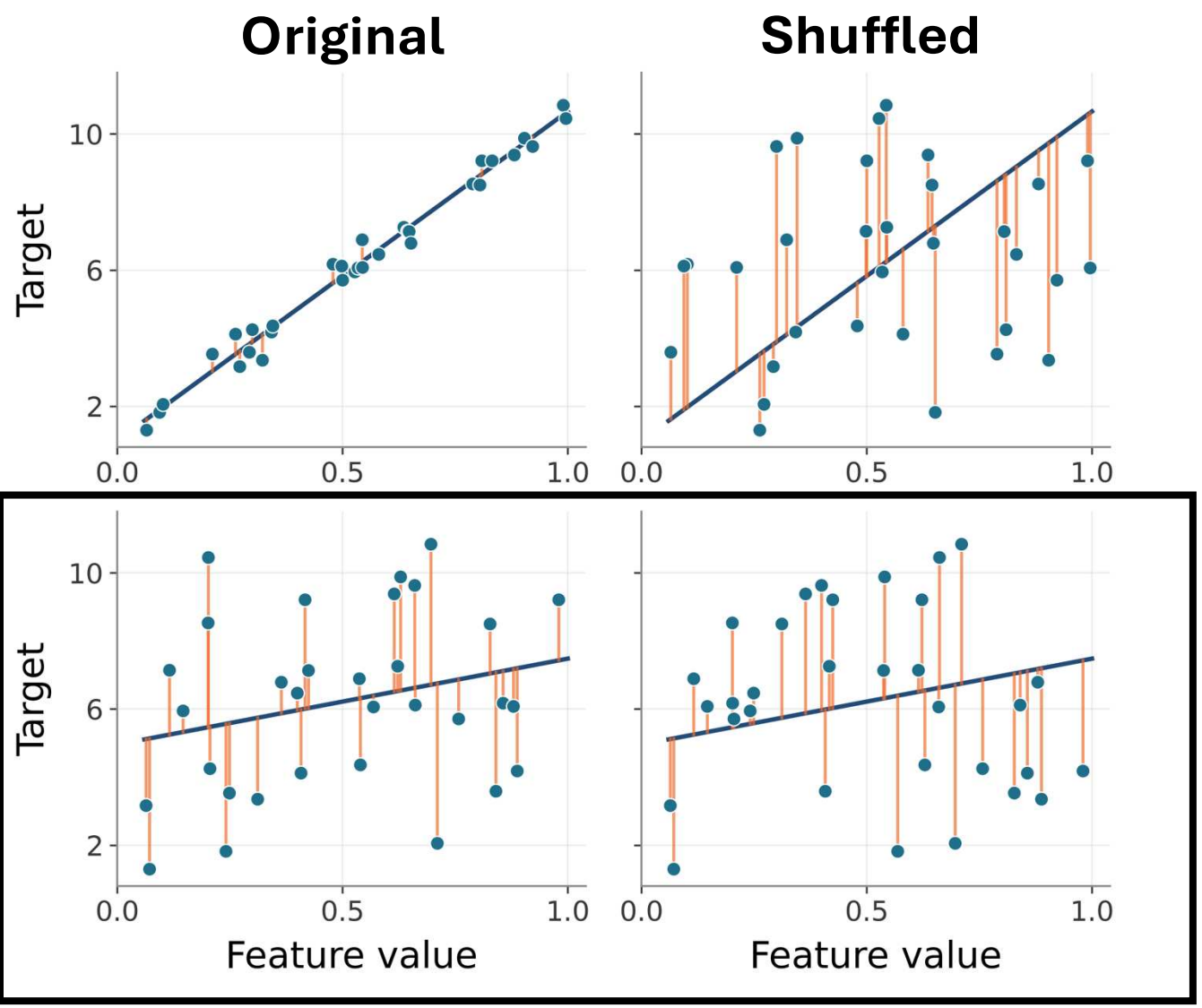
$$(x_1, y_1,), (x_2, y_2), \ldots, (x_n, y_n)$$

Shuffle the inputs, not the outputs

$$(x_n, y_1,), (x_1, y_2), \ldots, (x_2, y_n)$$

*Model can also be non-linear

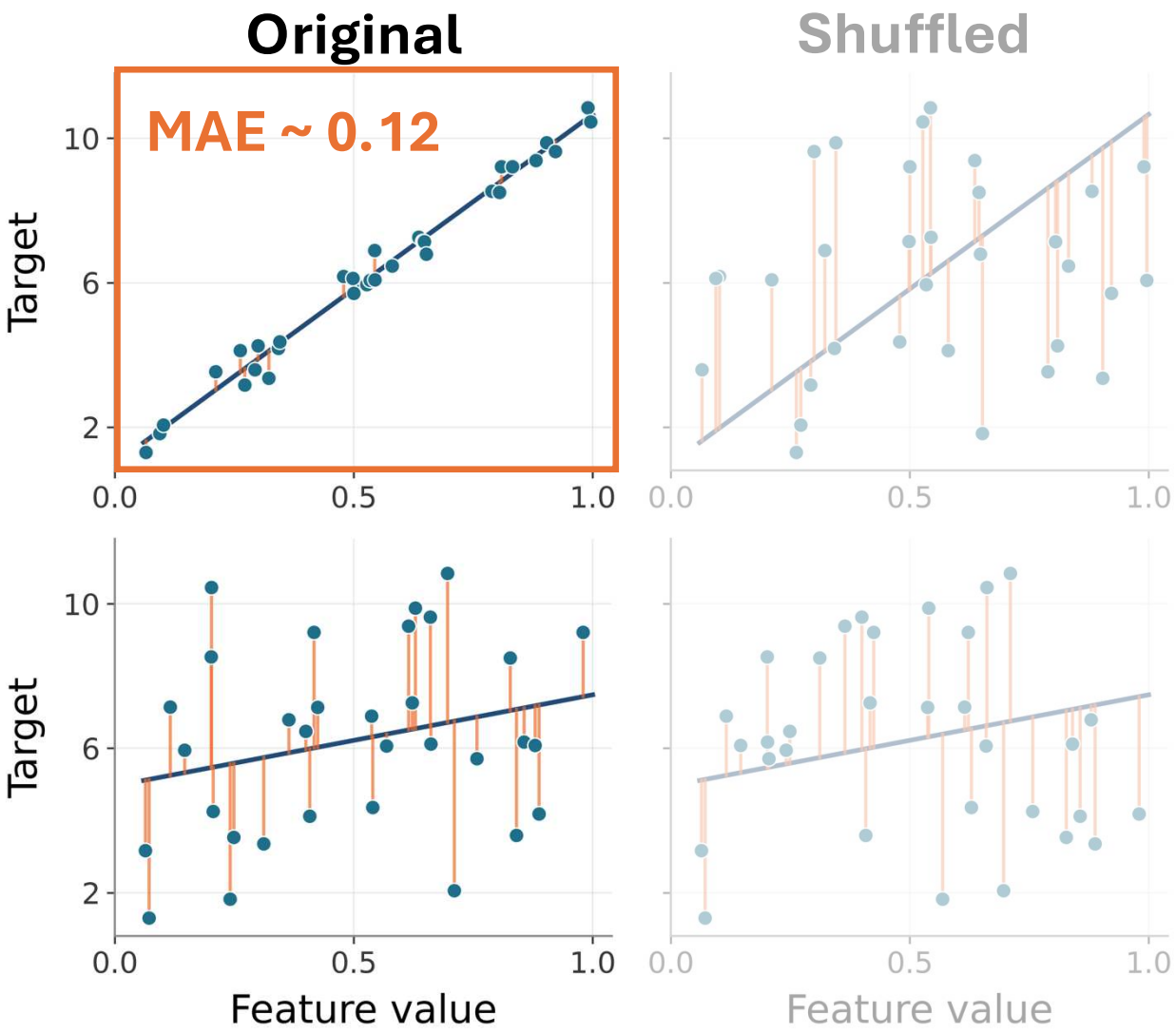
Permutation importance: General idea



Note that in practice, there are ***many input features***, and we will shuffle one at a time

Here, the feature was less important. It doesn't correlate much with the target. Our linear model doesn't suffer much if randomly swap inputs (x) while fixing the output (y).

Permutation importance: Step-by-step

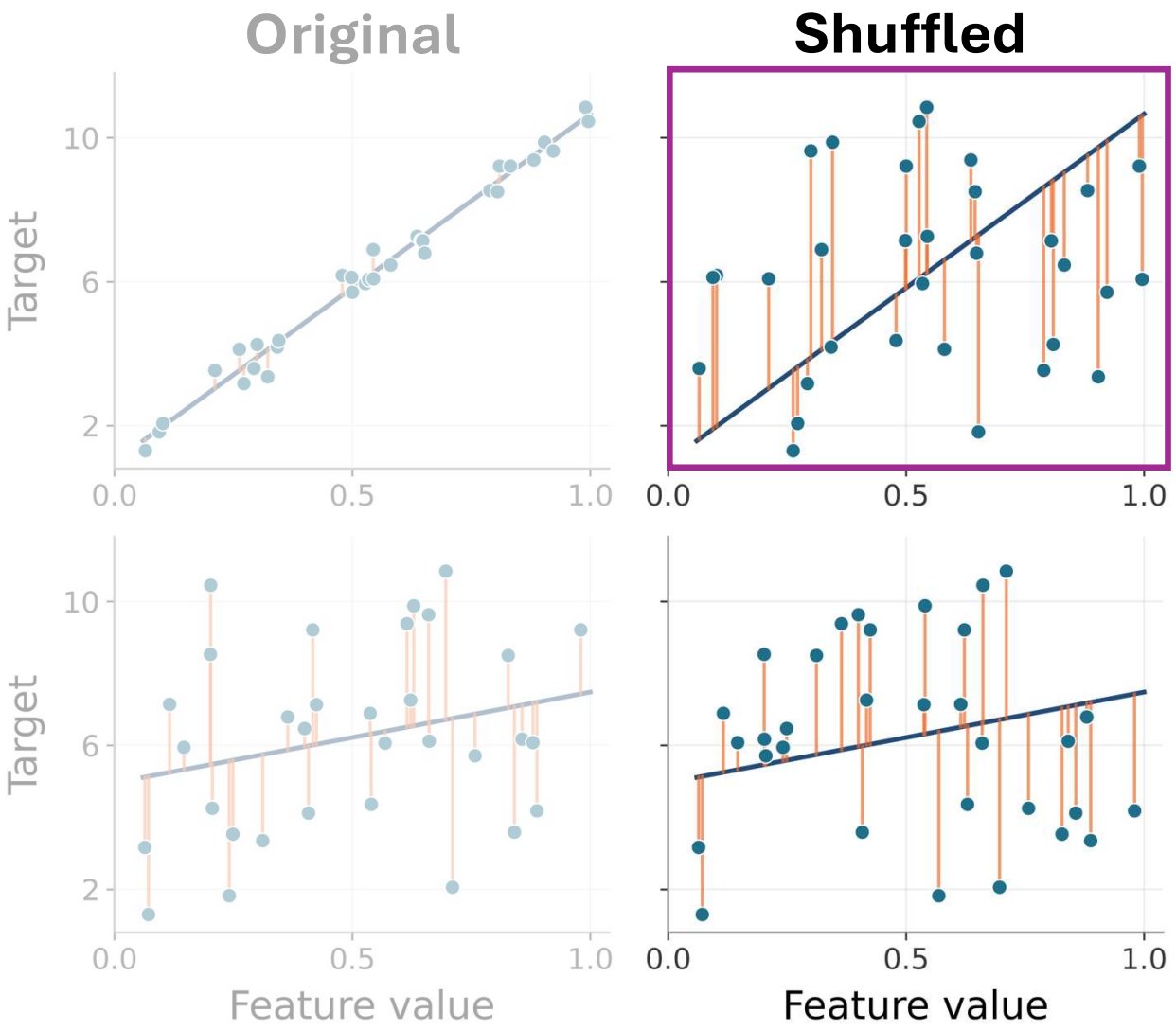


1. **Evaluate the model normally* and record its baseline performance**

2. **Select one feature (x_i) to shuffle**

*Usually, this is done on the validation set

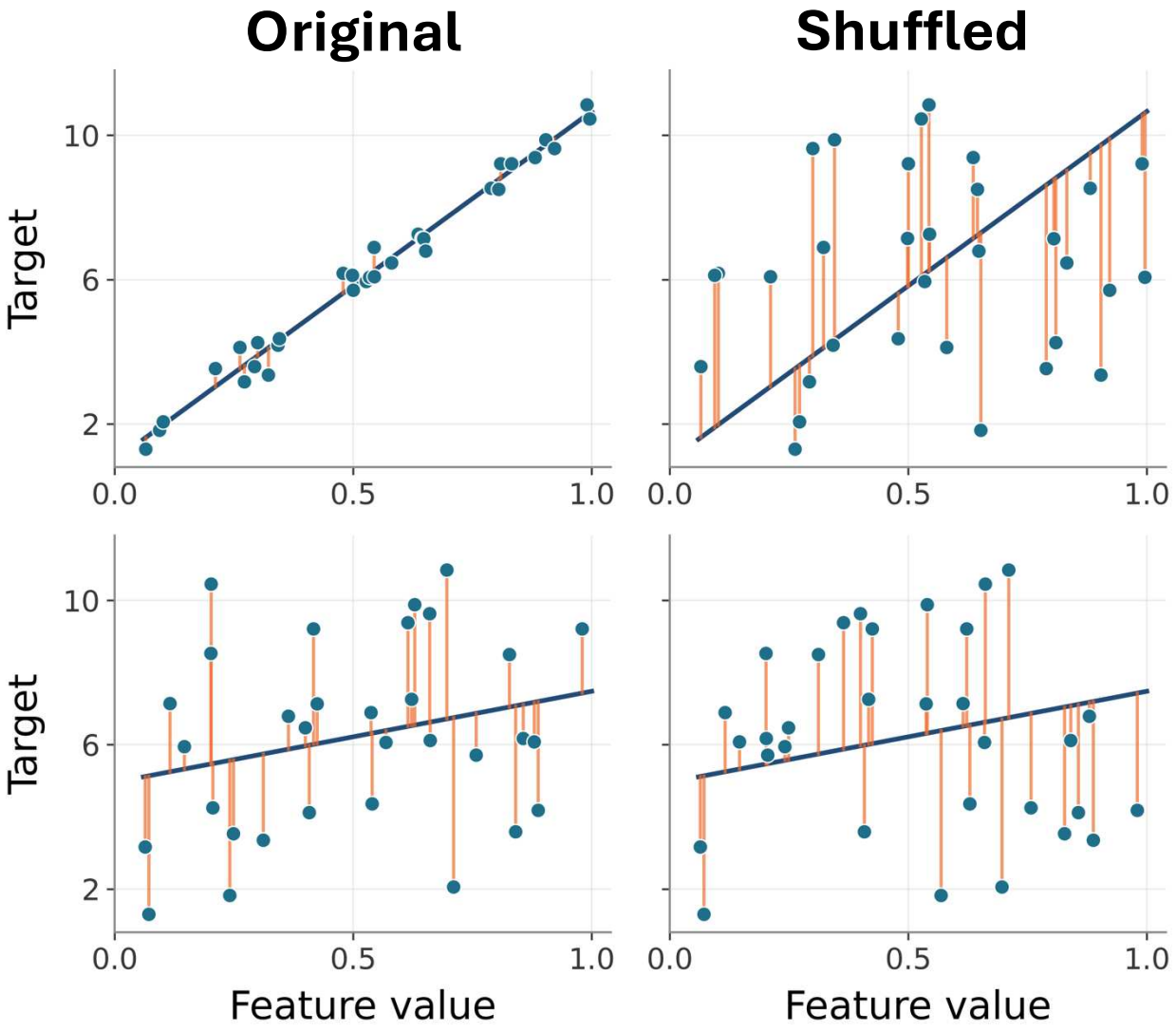
Permutation importance: Step-by-step

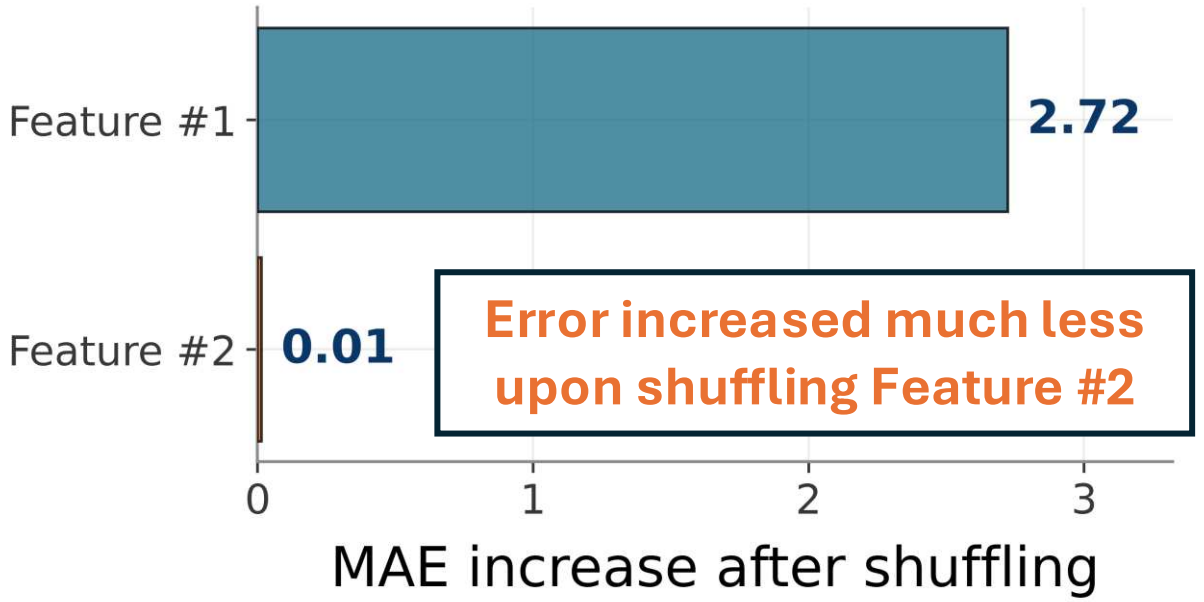


1. Evaluate the model normally* and record its baseline performance
2. Select one feature (x_i) to shuffle
3. Randomly shuffle x_i across samples while keeping all other features unchanged
4. Evaluate the same model again and measure how much the performance deteriorates

MAE 0.12 → 2.84 upon shuffling Feature #1

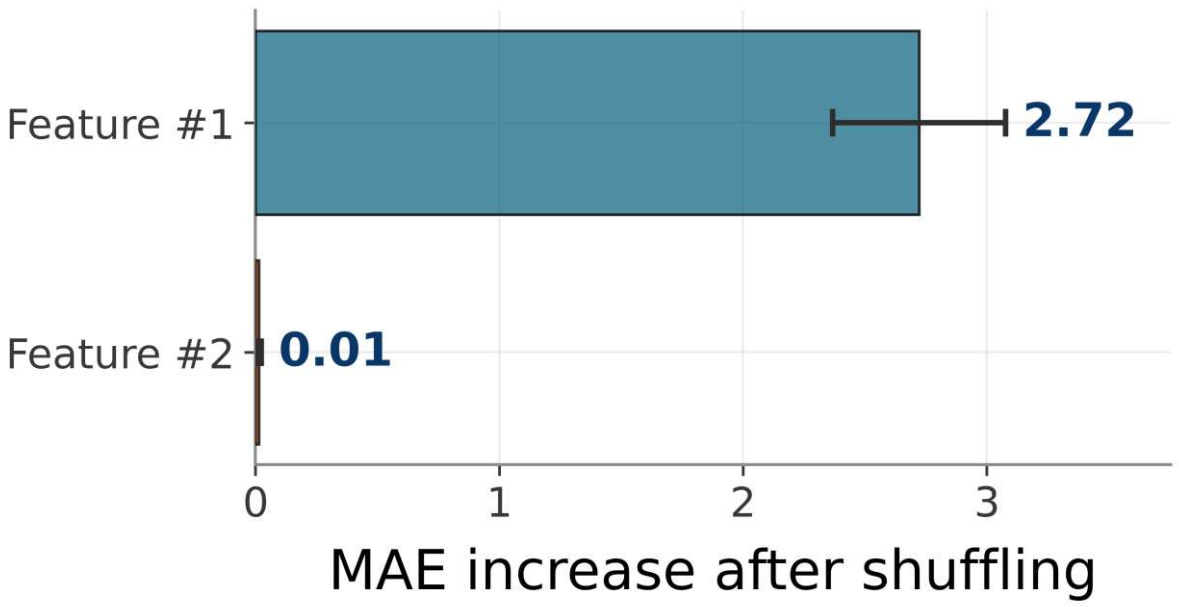
Permutation importance: Results

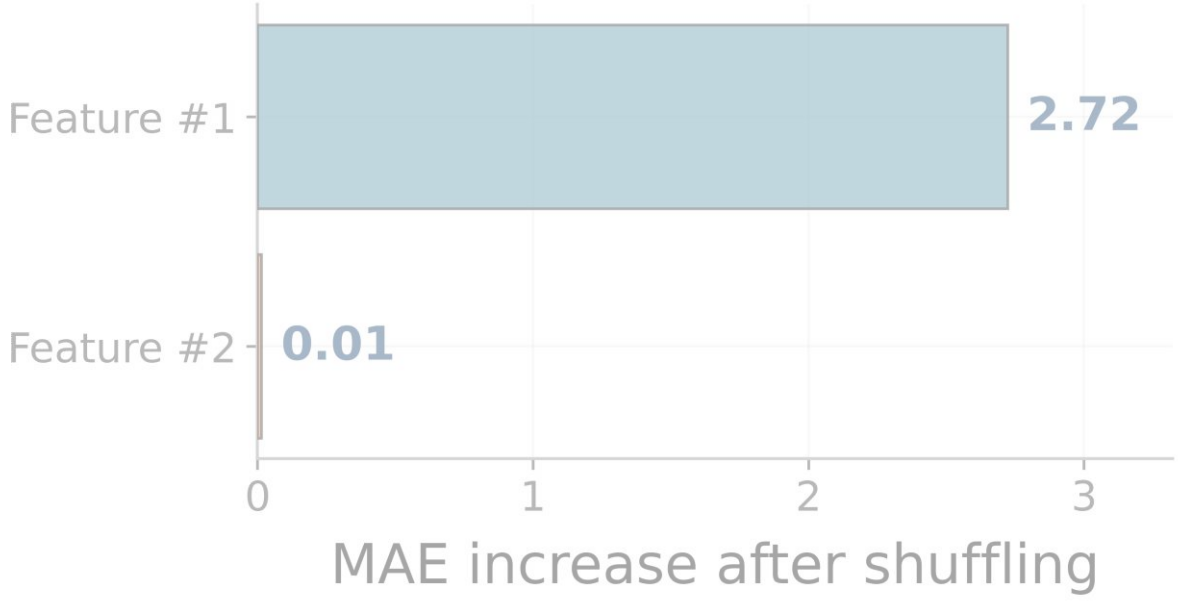




- Permutation importance has the **same units as the target**
- We use MAE here, but importance can be calculated using **any performance metric** (MSE, F1, cross-entropy, *etc.*)

Permutation importance: Results

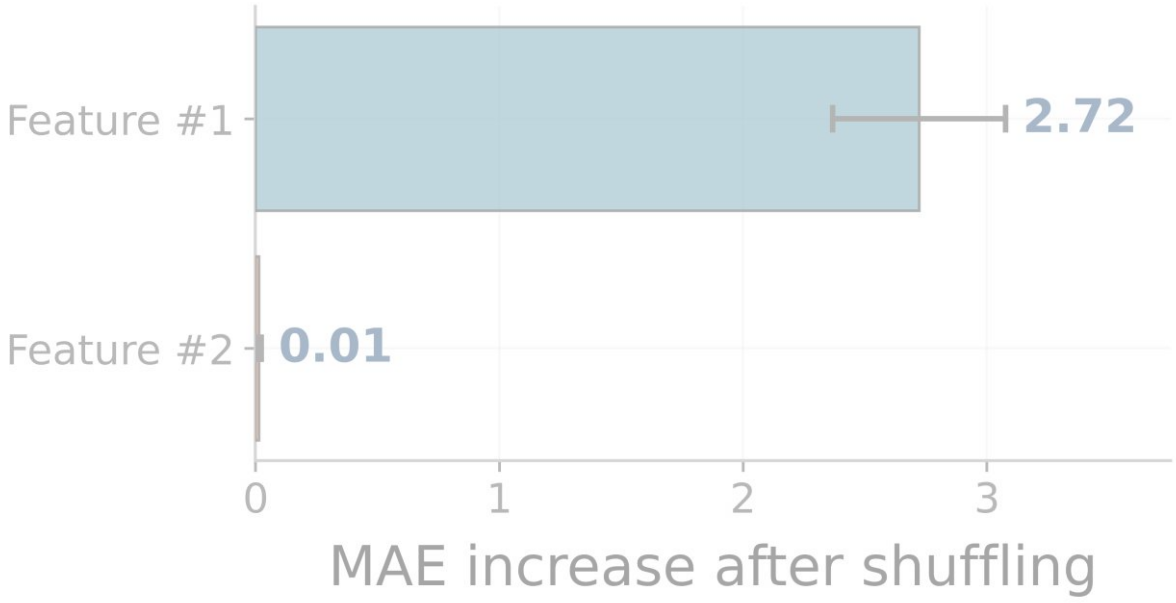




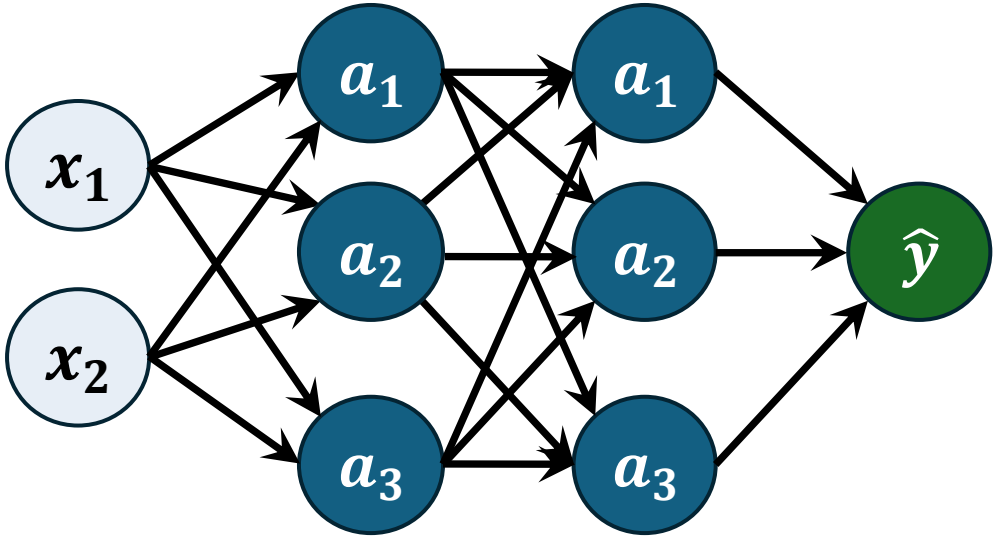
Because permutation is random, **each feature can be shuffled multiple times**. Bars then show the mean decrease in performance, and **error bars show the variation across shuffles.**

- Permutation importance has the **same units as the target**
- We use MAE here, but importance can be calculated using **any performance metric** (MSE, F1, cross-entropy, *etc.*)

Permutation importance: Results

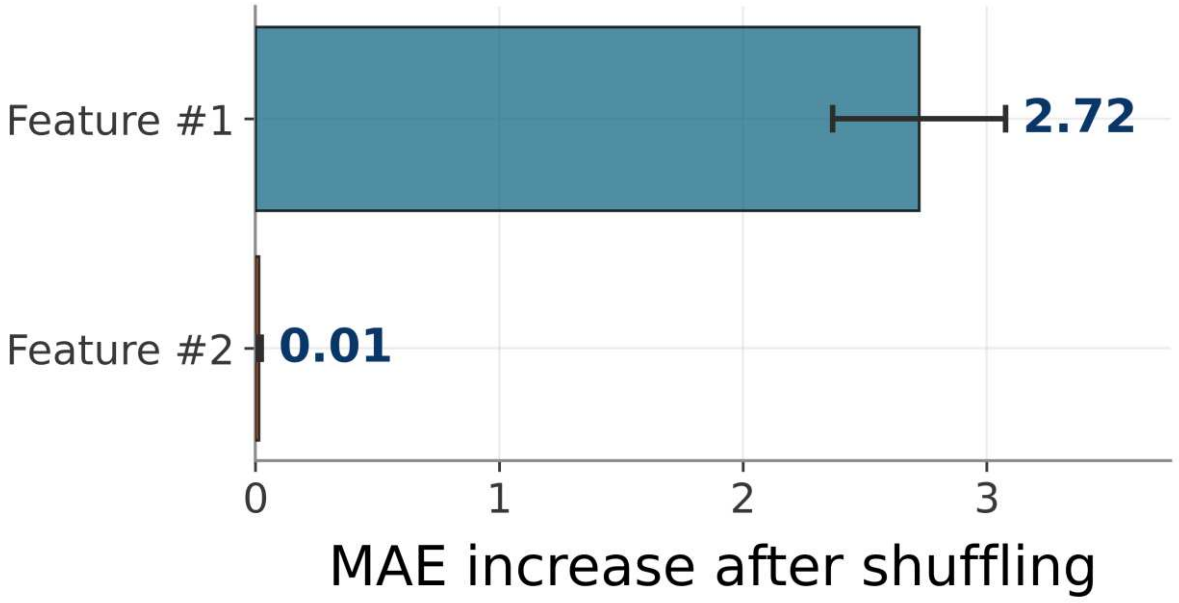


Because permutation is random, **each feature can be shuffled multiple times.** Bars then show the mean decrease in performance, and **error bars show the variation across shuffles.**



- The exact same procedure applies for neural networks (**model-agnostic**)
- Though whether it makes sense depends on the input type – for example, tabular data *vs.* images

Permutation importance: Limitations



Because permutation is random, **each feature can be shuffled multiple times.** Bars then show the mean decrease in performance, and **error bars show the variation across shuffles.**

Permutation importance tells us *which* features matter, but **not *how* they affect the prediction.**

We'd like to answer:

Does the target increase or decrease with Feature #1?

Is the relationship linear? Quadratic? Something else?

Partial dependence

Partial dependence is just how it sounds: How do the model predictions vary (**depend**) as we change just one feature (**partial** input)

Suppose $y = f(x_1, x_2, \dots, x_p)$ and we have **training data inputs:** $(x_{1,0}, x_{2,0}, \dots, x_{n,0})$

We could make predictions ($\hat{y}$) over these points, but instead, **choose one feature (x_j) to vary**

$(x_{1,1}, x_{2,1}, \dots, x_{n,1})$

$(x_{1,2}, x_{2,2}, \dots, x_{n,2})$

Then consider the **modified inputs** with **iteratively higher x_j:**

$(x_{1,0}, x_{2,0}, \dots, x_{j,0} = 0, \dots, x_{n,0})$

$(x_{1,1}, x_{2,1}, \dots, x_{j,1} = 0, \dots, x_{n,1})$

$(x_{1,1}, x_{2,1}, \dots, x_{j,2} = 0, \dots, x_{n,2})$

$(x_{1,0}, x_{2,0}, \dots, x_{j,0} = 0.1, \dots, x_{n,0})$

$(x_{1,1}, x_{2,1}, \dots, x_{j,1} = 0.1, \dots, x_{n,1})$

$(x_{1,1}, x_{2,1}, \dots, x_{j,2} = 0.1, \dots, x_{n,2})$

Each time, plug into the model and make a prediction

Partial dependence

The range of values you test will depend on the problem – for example:

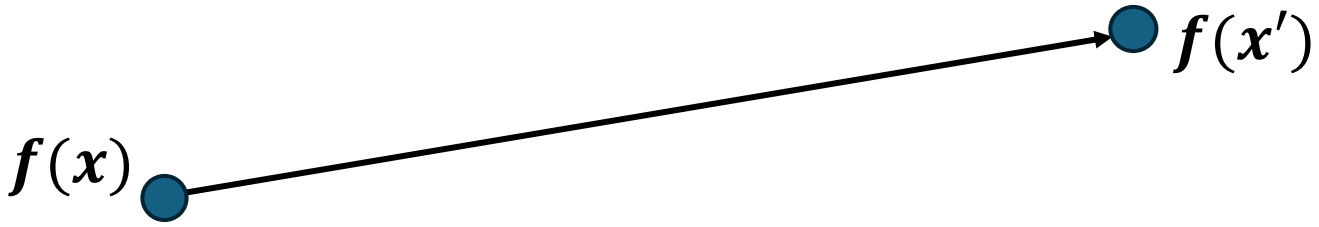
- Hours studied for an exam (x_j from 0 to 12 hours)
- Temperature of a chemical reaction (x_j from 25 to 1000 °C)

$$(x_{1,0}, x_{2,0}, \dots, x_{j,0} = 0, \dots, x_{n,0})$$
$$(x_{1,1}, x_{2,1}, \dots, x_{j,1} = 0, \dots, x_{n,1})$$
$$(x_{1,1}, x_{2,1}, \dots, x_{j,2} = 0, \dots, x_{n,2})$$

$$(x_{1,0}, x_{2,0}, \dots, x_{j,0} = 0.1, \dots, x_{n,0})$$
$$(x_{1,1}, x_{2,1}, \dots, x_{j,1} = 0.1, \dots, x_{n,1})$$
$$(x_{1,1}, x_{2,1}, \dots, x_{j,2} = 0.1, \dots, x_{n,2})$$

Each time, plug into the model and make a prediction

Partial dependence



Average model predictions over *all* samples*, for a fixed x_j

$$(x_{1,0}, x_{2,0}, \dots, x_{j,0} = 0, \dots, x_{n,0})$$
$$(x_{1,1}, x_{2,1}, \dots, x_{j,1} = 0, \dots, x_{n,1})$$
$$(x_{1,1}, x_{2,1}, \dots, x_{j,2} = 0, \dots, x_{n,2})$$

Then do it again for a slightly larger x_j

$$(x_{1,0}, x_{2,0}, \dots, x_{j,0} = 0.1, \dots, x_{n,0})$$
$$(x_{1,1}, x_{2,1}, \dots, x_{j,1} = 0.1, \dots, x_{n,1})$$
$$(x_{1,1}, x_{2,1}, \dots, x_{j,2} = 0.1, \dots, x_{n,2})$$

And so on…
The trend reveals how x_j affects the model output

*In training or validation set

Partial dependence: Math

Partial dependence asks: if I dial this feature up, what happens on average?

$$PD_j(z) = \frac{1}{N}\sum_{n=1}^{N} f(x_j = z, x_{n,\text{others}})$$

The partial dependence of feature j at value z

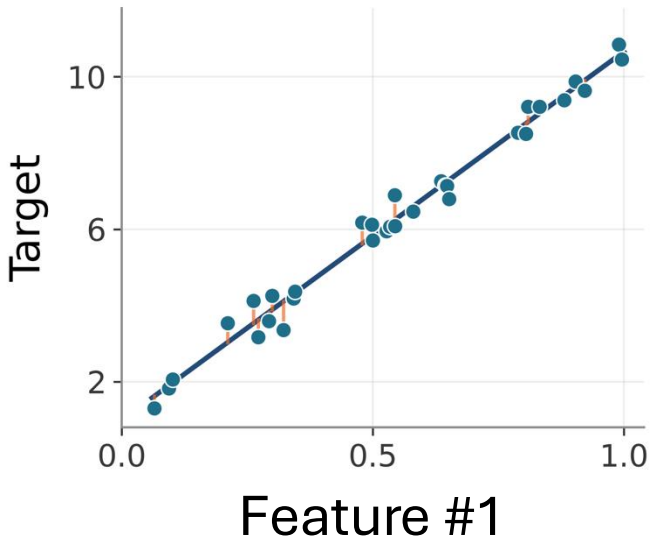
Sum over all samples in the dataset

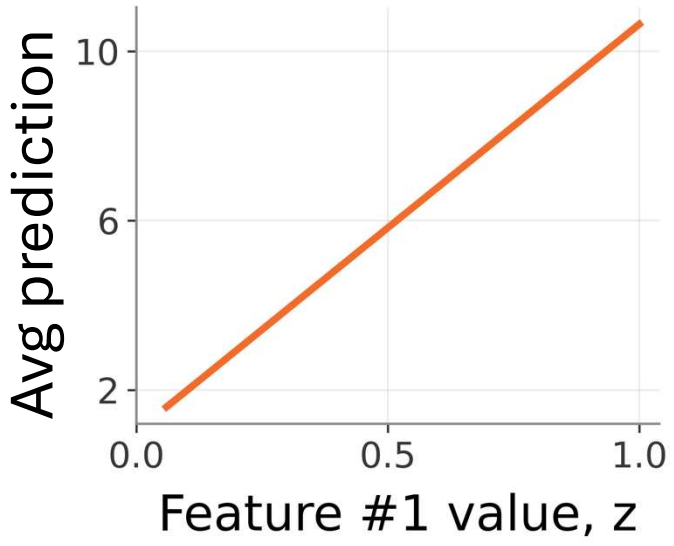
Plugged into our model

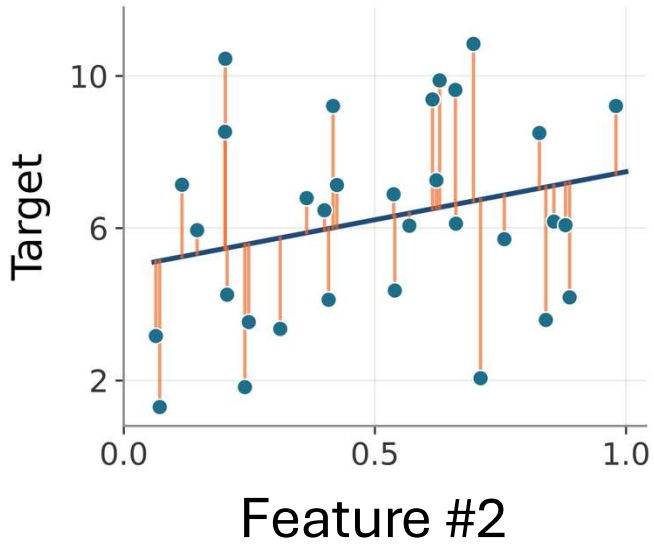
Keep feature j fixed at value z

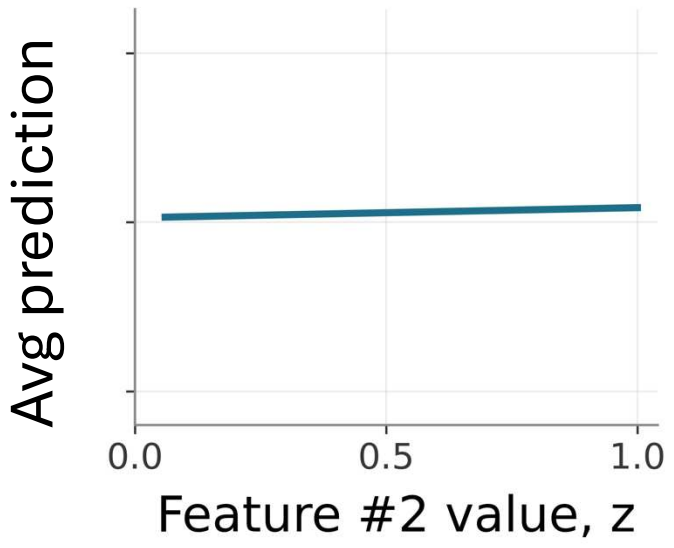
Calculate this over a range of z values for feature j → yielding a curve!

Partial dependence plots (PDPs)





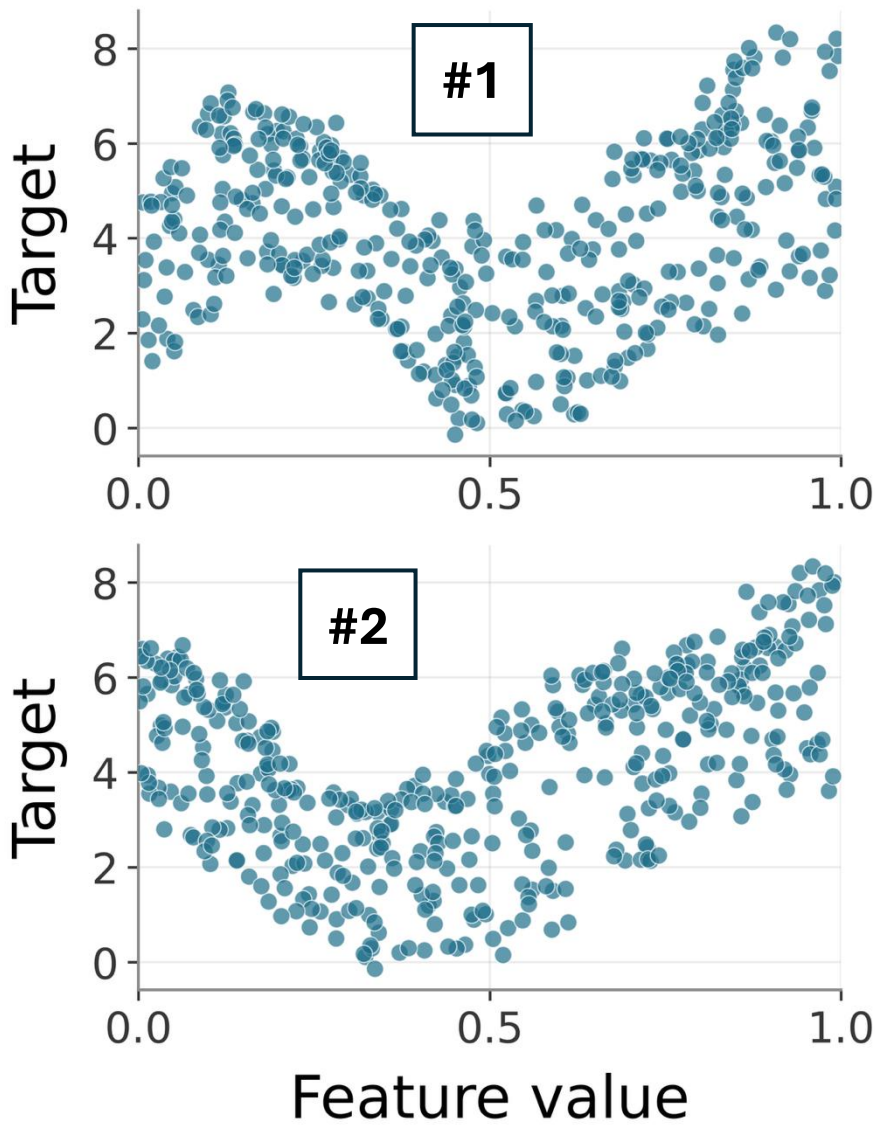


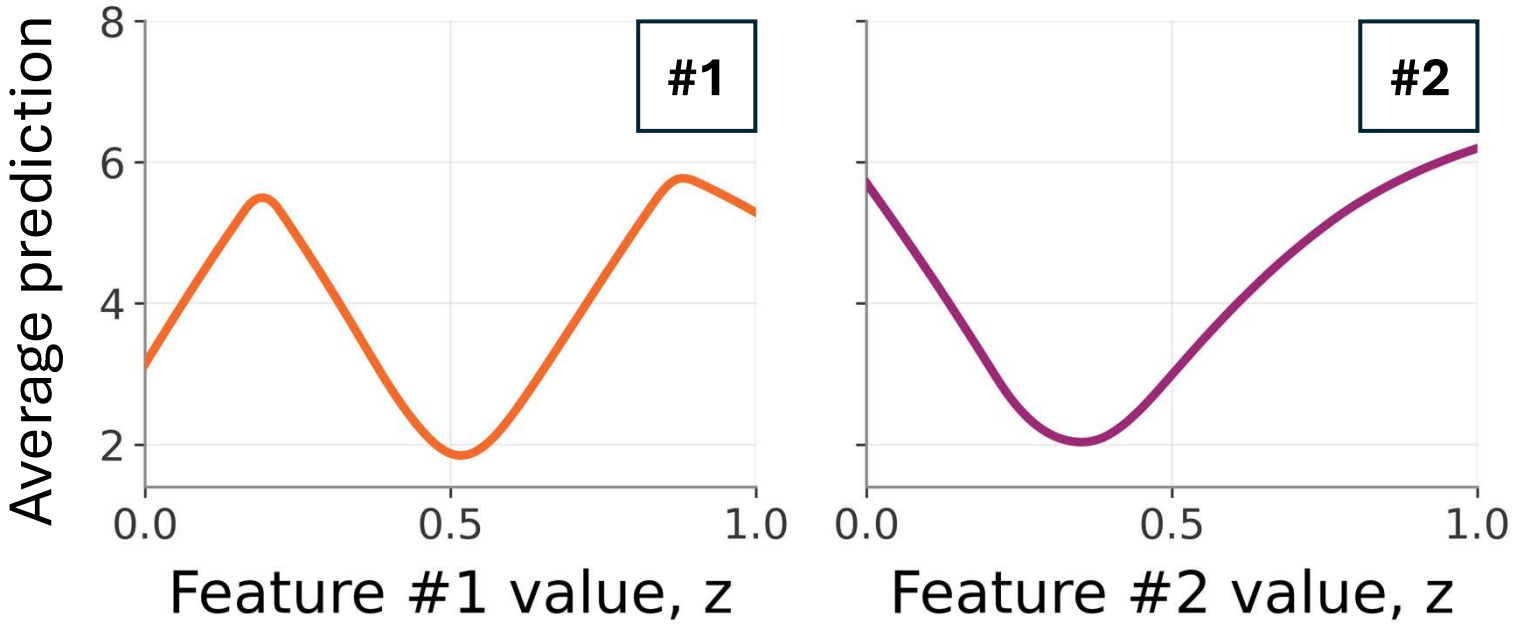


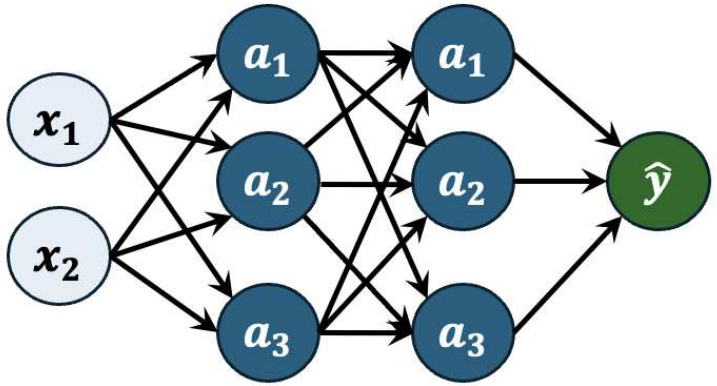
Since our model is linear, so is the PDP. In this case, the plot is a bit “boring”

But **PDPs can be applied to *any* model!**

Partial dependence plots (PDPs)

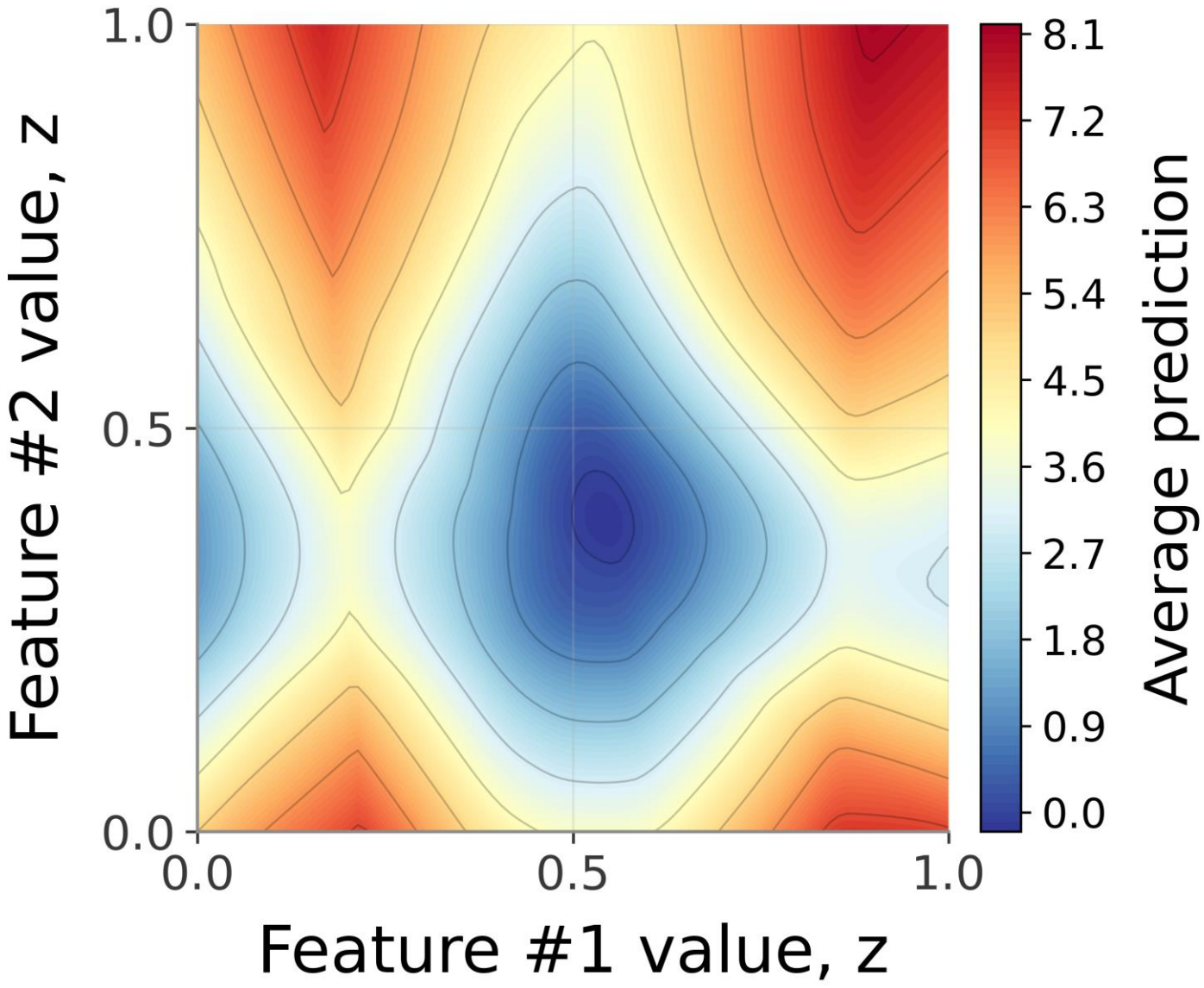






This often becomes more useful when there are **many descriptors** that obscure clear trends in raw data

PDPs can also be plotted in 2D

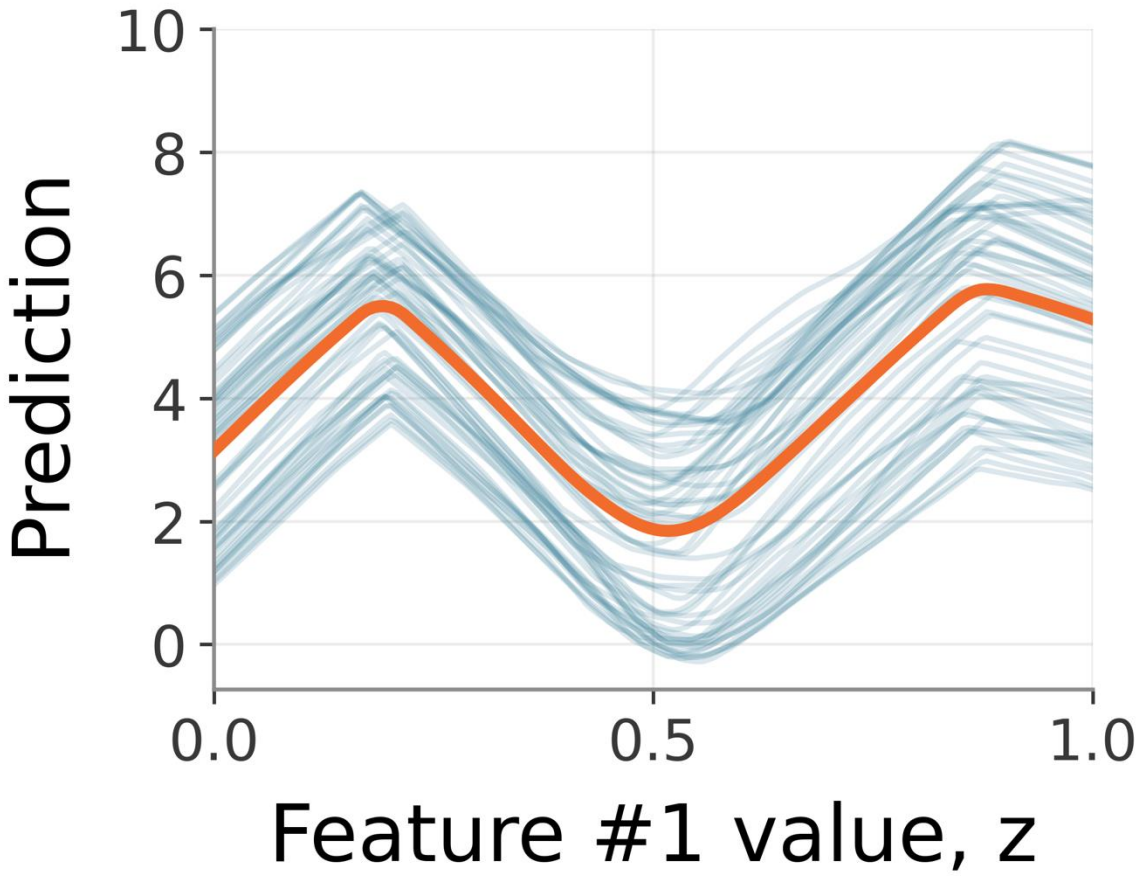


Shows joint effects by revealing how two features influence the model's prediction when varied together.

Reveals interactions by showing whether the effect of Feature #1 changes depending on the value of Feature #2.

If we only have two features, this is just the model's output over a grid. Becomes more useful with > 2 features!

Individual Conditional Expectation (ICE) plots



Partial dependence asks: if I dial this feature up, what happens **on average?**

$$PD_j(z) = \frac{1}{N}\sum_{n=1}^{N} f(x_j = z, x_{n,\text{others}})$$

Each partial dependence value is an average over *all* samples…

We can also calculate $PD_j(z)$ for each sample, without averaging

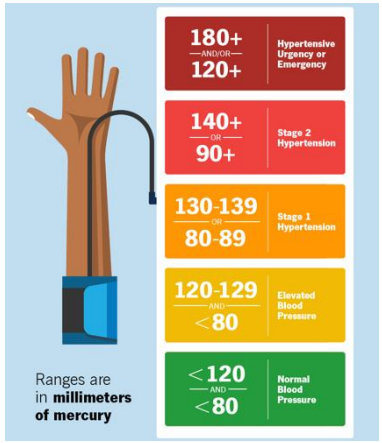
ICE plots can be helpful to see variations in the data
Large variations → the effect of x_j varies from sample to sample

Questions?

Back to our table: What about local explanations?

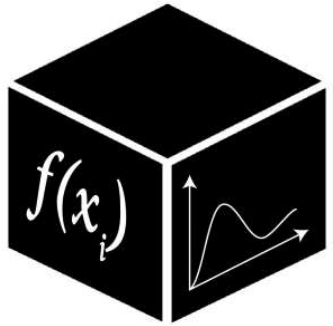
	Model-specific	Model-agnostic
Global	Regression coefficients (β_i) Odds ratios (e^{β_i}/e^{β_j}) Feature importance based on information gain in trees	**Today: Partial dependence, permutation importance, and ICE curves**
Local	**Why did the model make a specific prediction?**	**Today: LIME and SHAP**

Which features contribute to a prediction?

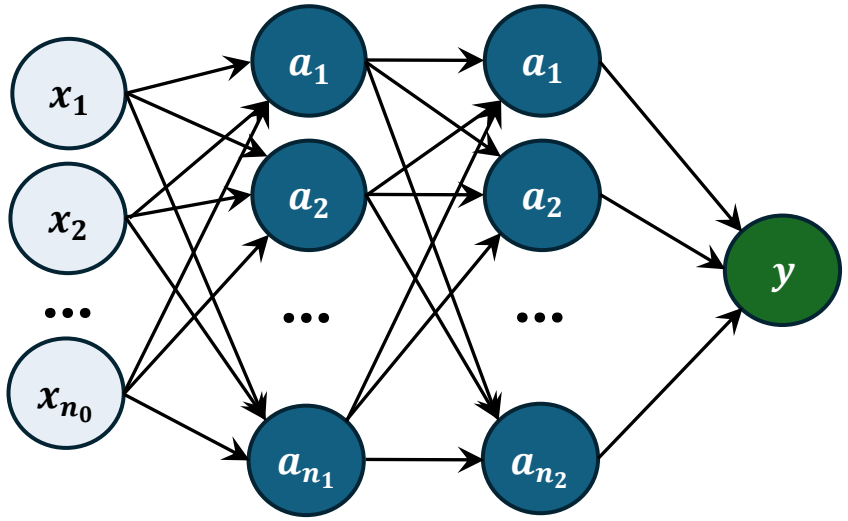








$$P(\text{operate}|x)$$



Given one sample, our model suggests:
operate with a probability of 84%

Which feature(s) most "pushed" this probability up or down? By how much?

With a NN, the relationship between input/output is highly nonlinear…
We don't have simple coefficients

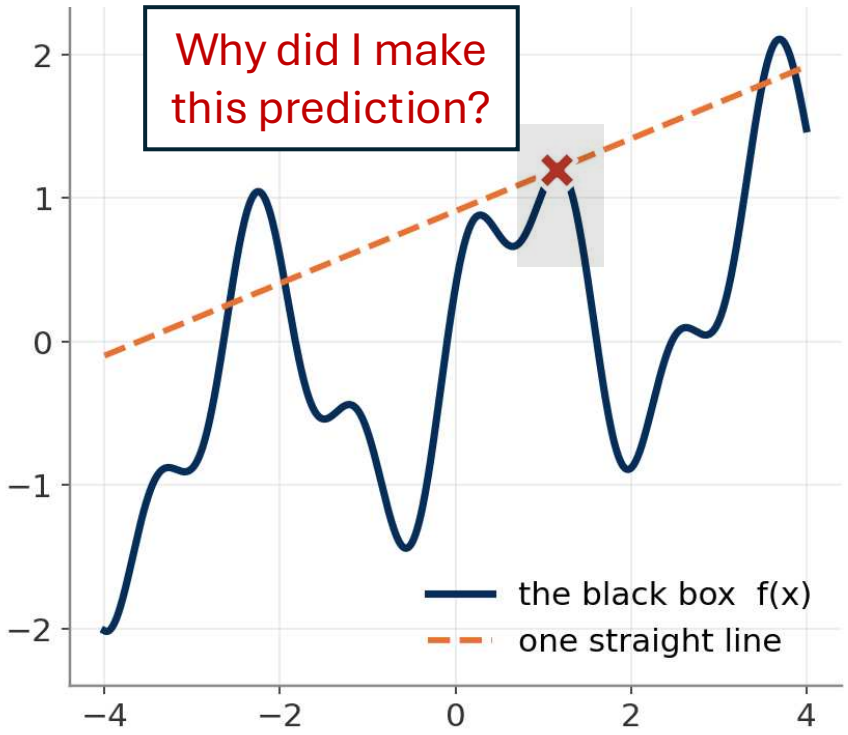
Local, Interpretable, Model-agnostic Explanations

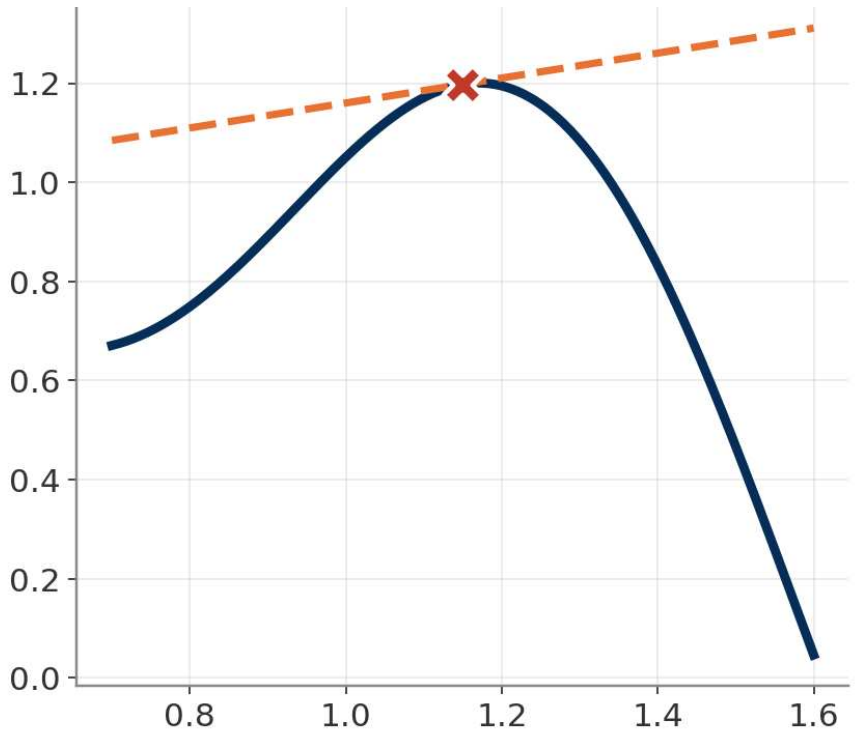
"Why Should I Trust You?"
Explaining the Predictions of Any Classifier

Marco Tulio Ribeiro Sameer Singh Carlos Guestrin

→ **LIME**

9 Aug 2016

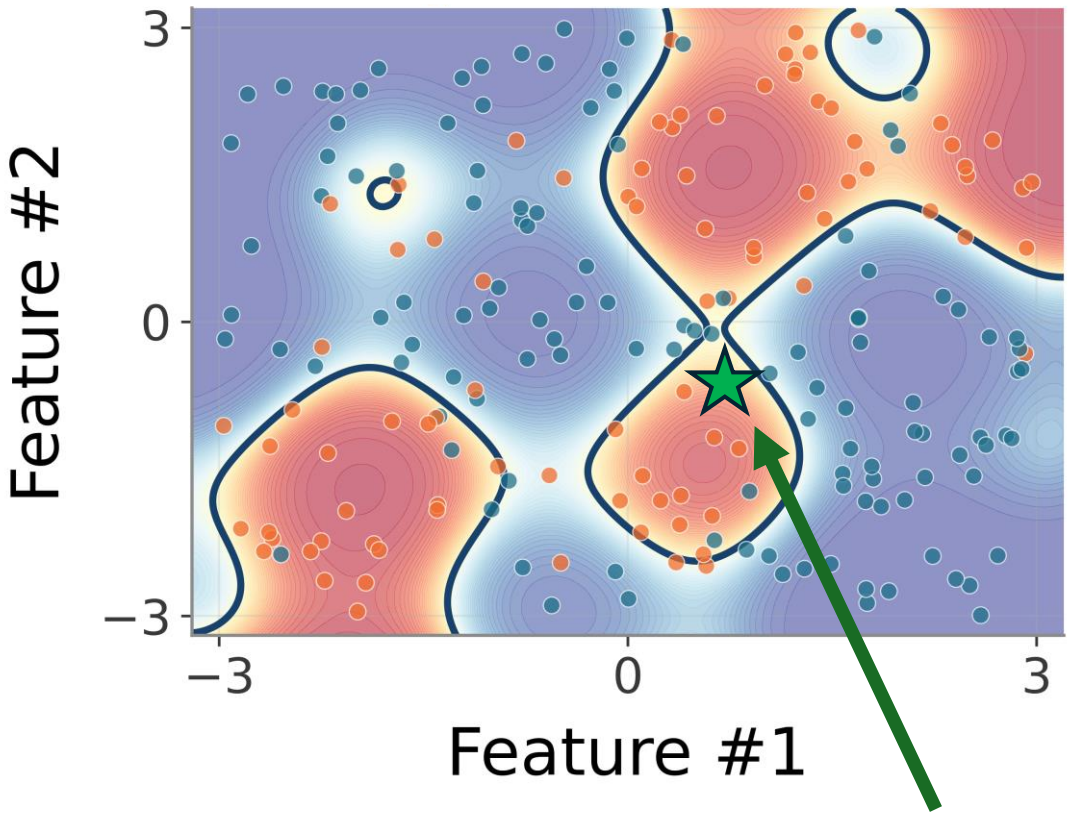




If you zoom in far enough, any function can be approximated by a straight line…

And linear models have coefficients that are nice and interpretable!

LIME: Step-by-step

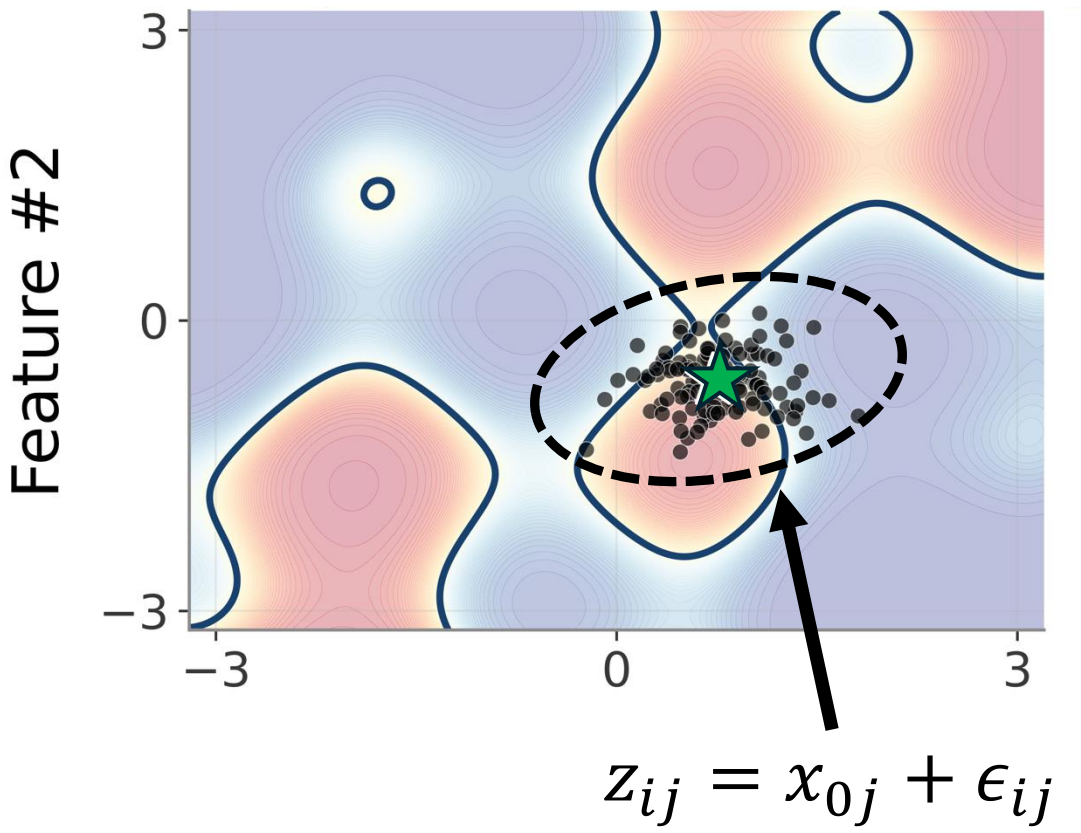


Why was this sample predicted Class #1?

Starting with a trained model:

1. **Choose the point to explain (x_0)**
2. Generate nearby samples (z_i)
3. Weight nearby samples (w_i)
4. Locally fit a linear model
5. Interpret the coefficients

LIME: Step-by-step



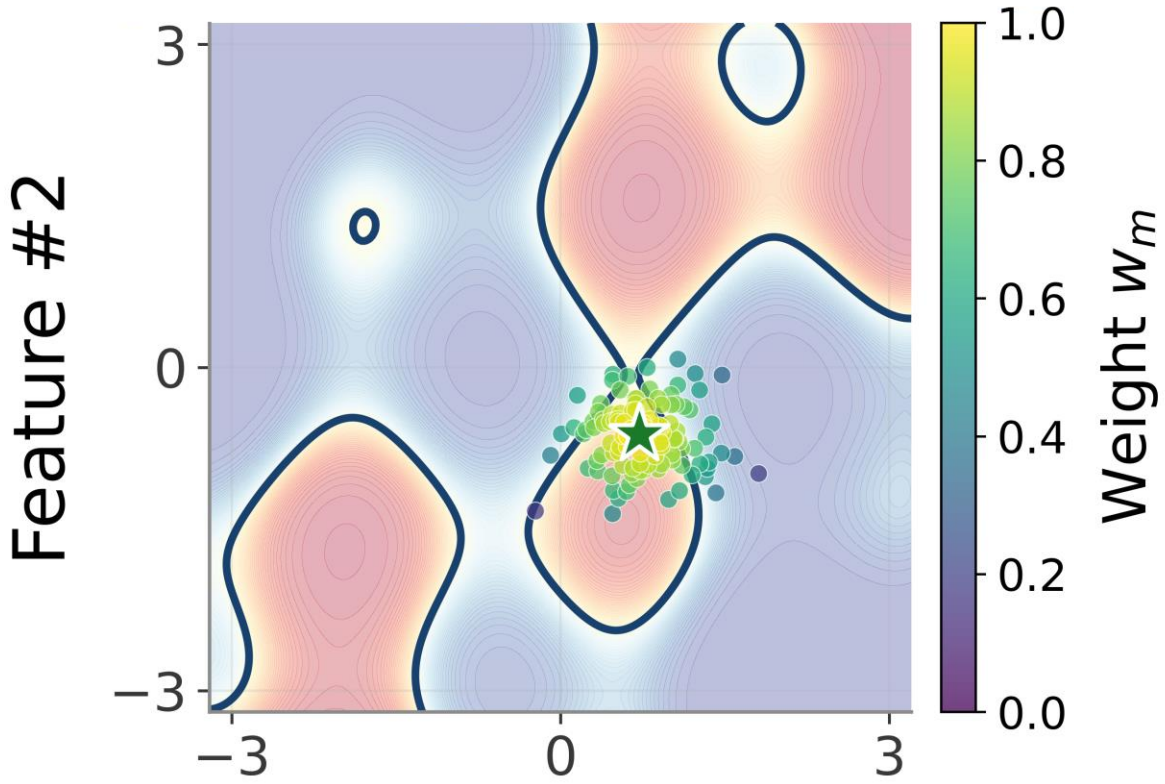
Usually, a few hundred to a few thousand points with $\epsilon_{ij} \sim 5 - 20\%$ each feature's range.

Calculate $y = f(z_{ij})$ where f is the model. ⟶ **From this, we have a new set of inputs (z) and outputs (y)**

Starting with a trained model:

1. Choose the point to explain (x_0)
2. **Generate nearby samples (z_i)**
3. Weight nearby samples (w_i)
4. Locally fit a linear model

LIME: Step-by-step



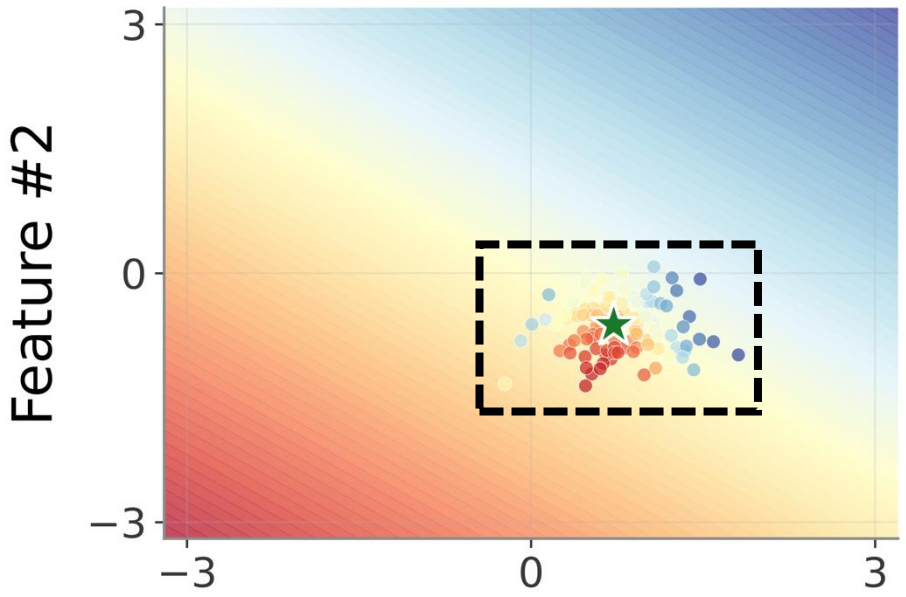
Starting with a trained model:

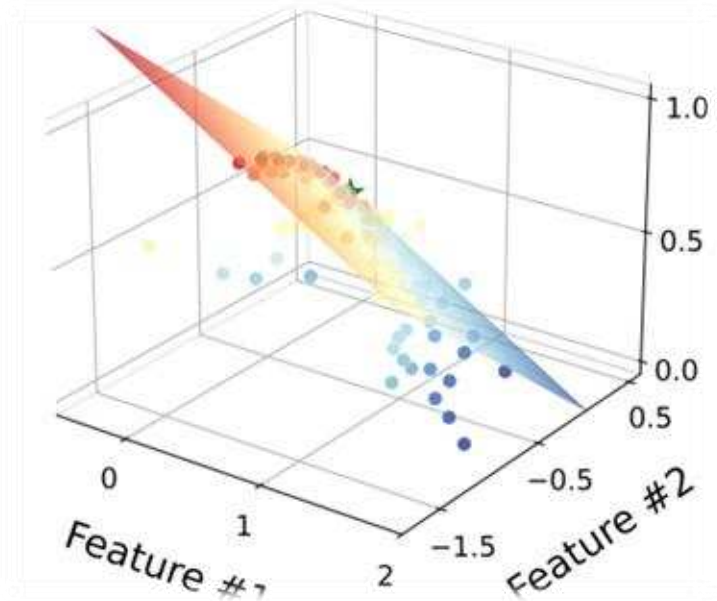
1. Choose the point to explain (x_0)
2. Generate nearby samples (z_i)
3. **Weight nearby samples (w_i)**
4. Locally fit a linear model
5. Interpret the coefficients

Nearby samples should matter more than faraway ones.

$$w_i = \exp\left[-\frac{d(z_i, x_0)}{\sigma^2}\right]$$

LIME: Step-by-step





This plane is reasonably valid in a close neighborhood around the point x_0

Fit: $g(z) = \beta_0 + \beta_1 z_1 + \cdots + \beta_p z_p$

By minimizing a **weighted loss:**

$$\mathcal{L} = \sum_i w_i [f(z_i) - g(z_i)]^2$$

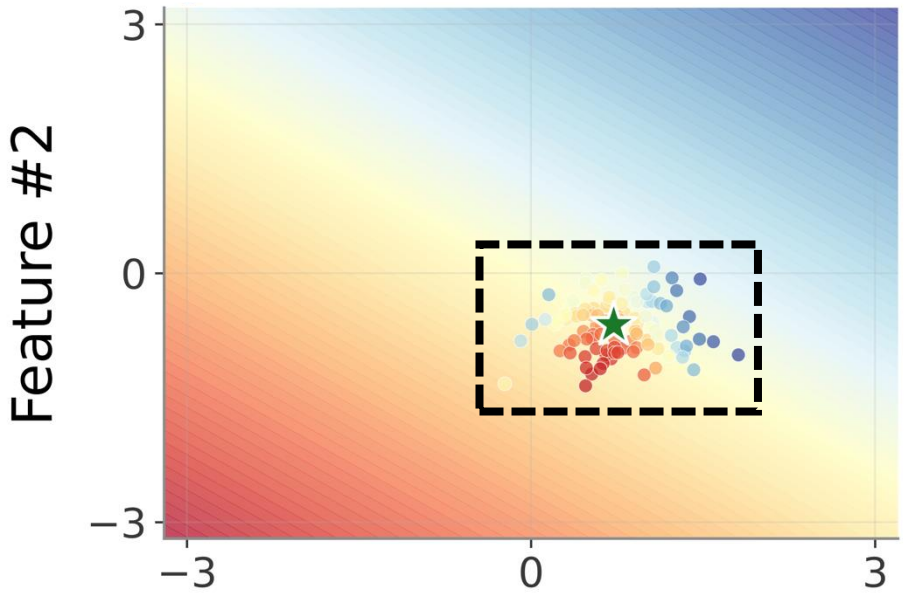
Black-box model's (non-linear) prediction

Linear model's prediction

4. **Locally fit a linear model**

5. Interpret the coefficients

LIME: Step-by-step



Fit: $g(z) = \beta_0 + \beta_1 z_1 + \cdots + \beta_p z_p$

By minimizing a **weighted loss:**

$$\mathcal{L} = \sum_i w_i [f(z_i) - g(z_i)]^2$$

Black-box model's (non-linear) prediction

Linear model's prediction

Interpret it as would any linear model:

$\beta_i > 0 \rightarrow$ Feature i pushes the target (probability) up

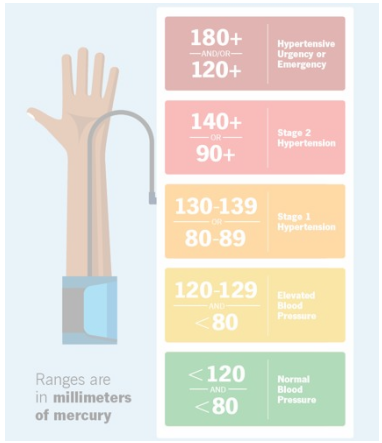
$\beta_i < 0 \rightarrow$ Feature i pushes the target (probability) down

$\beta_i = 0 \rightarrow$ Feature i doesn't matter

4. Locally fit a linear model

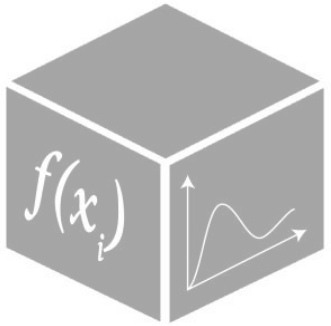
5. **Interpret the coefficients**

Visualizing LIME results

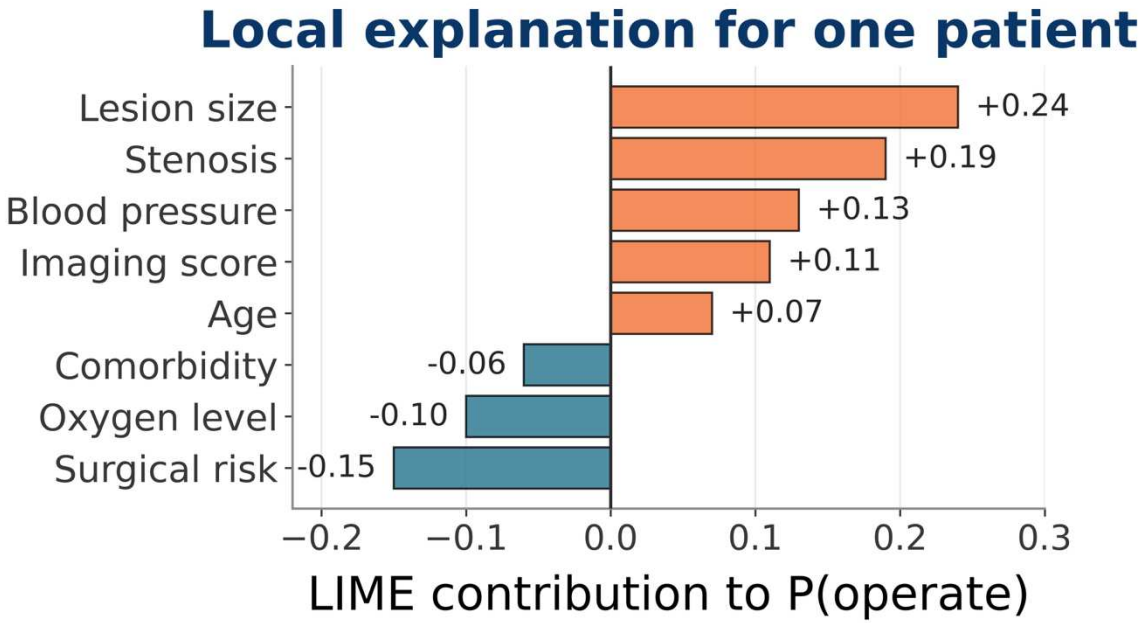








$P(\text{operate}|x)$



Given a sample, our model suggests:
operate with a probability of 84%

Which feature(s) most "pushed" this probability up or down? By how much?

Now we have coefficients to read! Clearly, lesion size is contributing strongly to the "operate" decision

Questions?

How about LIME for *image* classification?

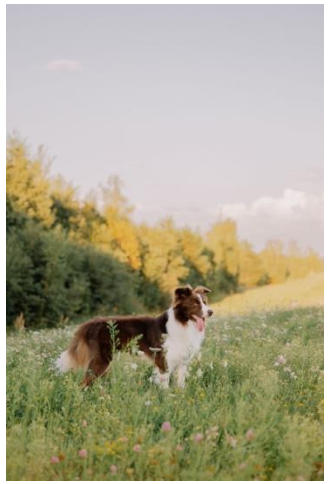
Wolves

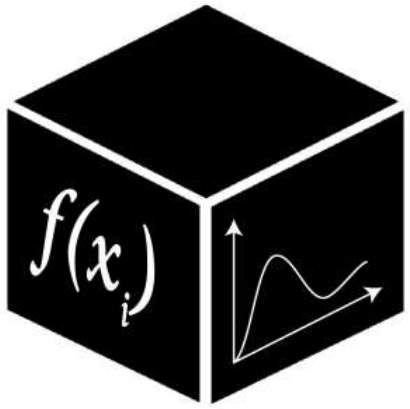





Dogs







With interpretable predictions, we figure out that our model uses the background (white *vs*. green) to decide

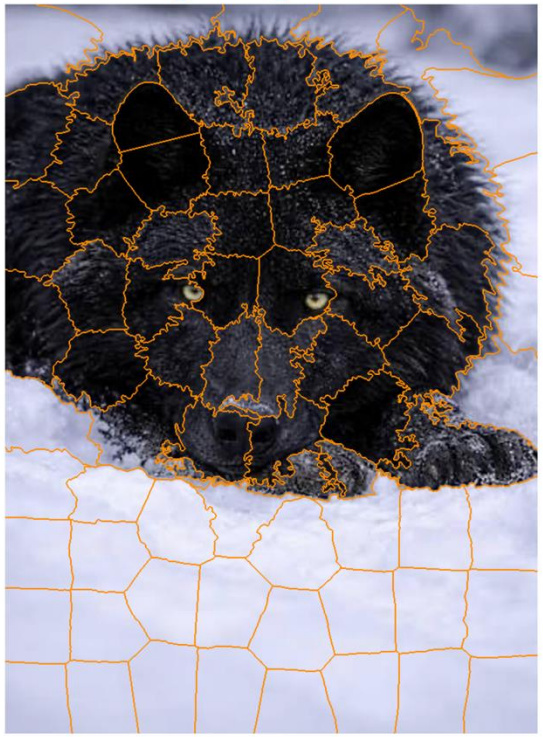
"I used this part of the image to classify it as a wolf!"

LIME for image classification: step-by-step

Original image



Superpixels

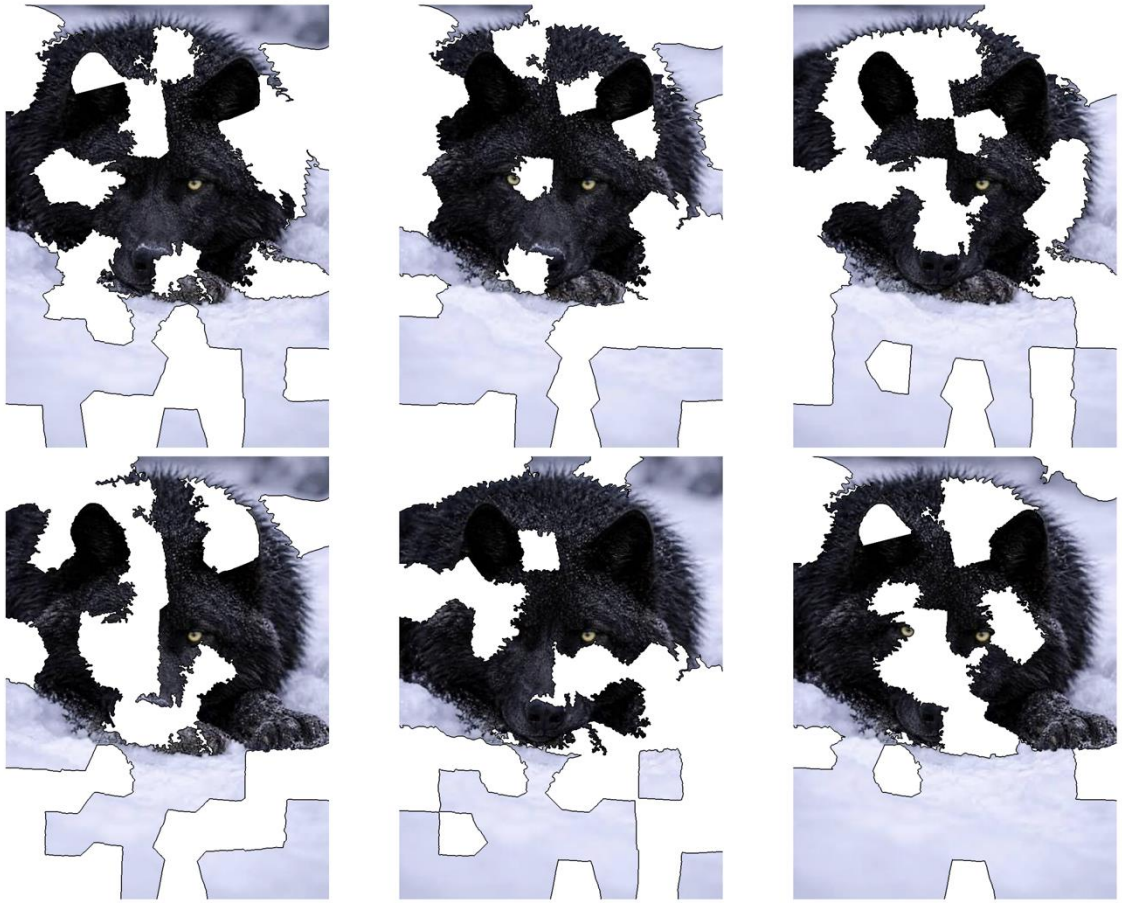


This can be done using a variety of segmentation algorithms that group based on pixel similarity

Starting with a trained model:

1. **Divide the image into *superpixels***
2. Created perturbed images
3. Weight nearby samples (w_i)
4. Locally fit a linear model
5. Interpret the coefficients

LIME for image classification: step-by-step

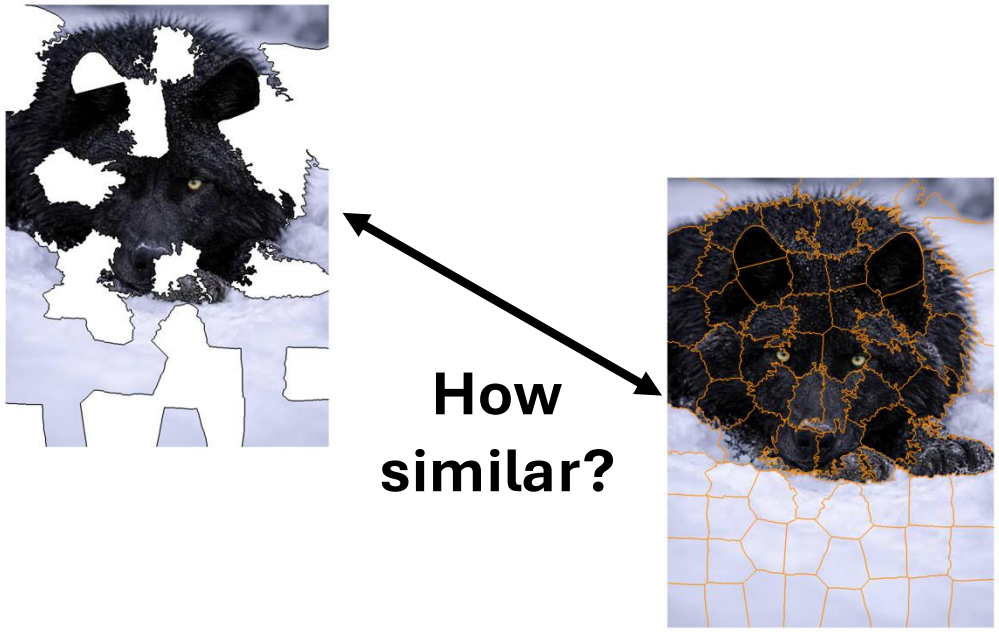


Create each by randomly “turning off” some fraction of the superpixels. Then make a prediction by passing it to the model.

Starting with a trained model:

1. Divide the image into *superpixels*
2. **Created perturbed images**
3. Weight nearby samples (w_i)
4. Locally fit a linear model
5. Interpret the coefficients

LIME for image classification: step-by-step



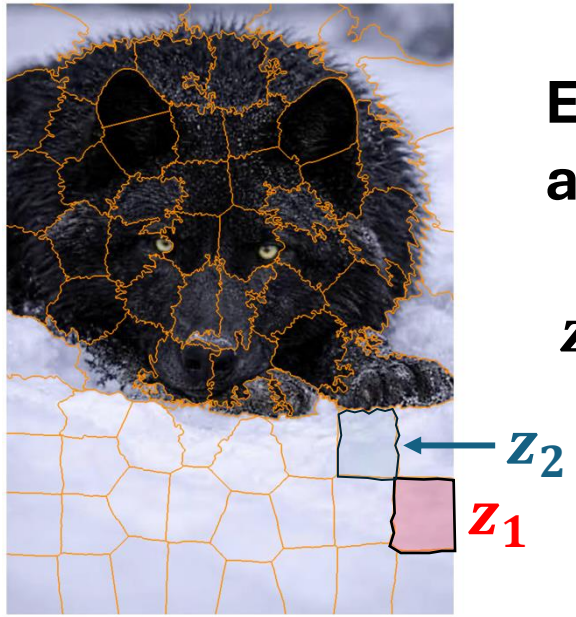
Similarity is based on how many superpixels remain the same as in the original image.

More superpixels removed = give it less weight

Starting with a trained model:

1. Divide the image into *superpixels*
2. Created perturbed images
3. **Weight nearby samples (w_i)**
4. Locally fit a linear model
5. Interpret the coefficients

LIME for image classification: step-by-step



Each superpixel is a one-hot feature:

$$z_k = \begin{cases} 1, \text{if present} \\ 0, \text{if not} \end{cases}$$

Fit: $g(z) = \beta_0 + \beta_1 z_1 + \cdots + \beta_p z_p$

By minimizing a **weighted loss:**

$$\mathcal{L} = \sum_i w_i [f(z_i) - g(z_i)]^2$$

Black-box model's (non-linear) prediction

Linear model's prediction

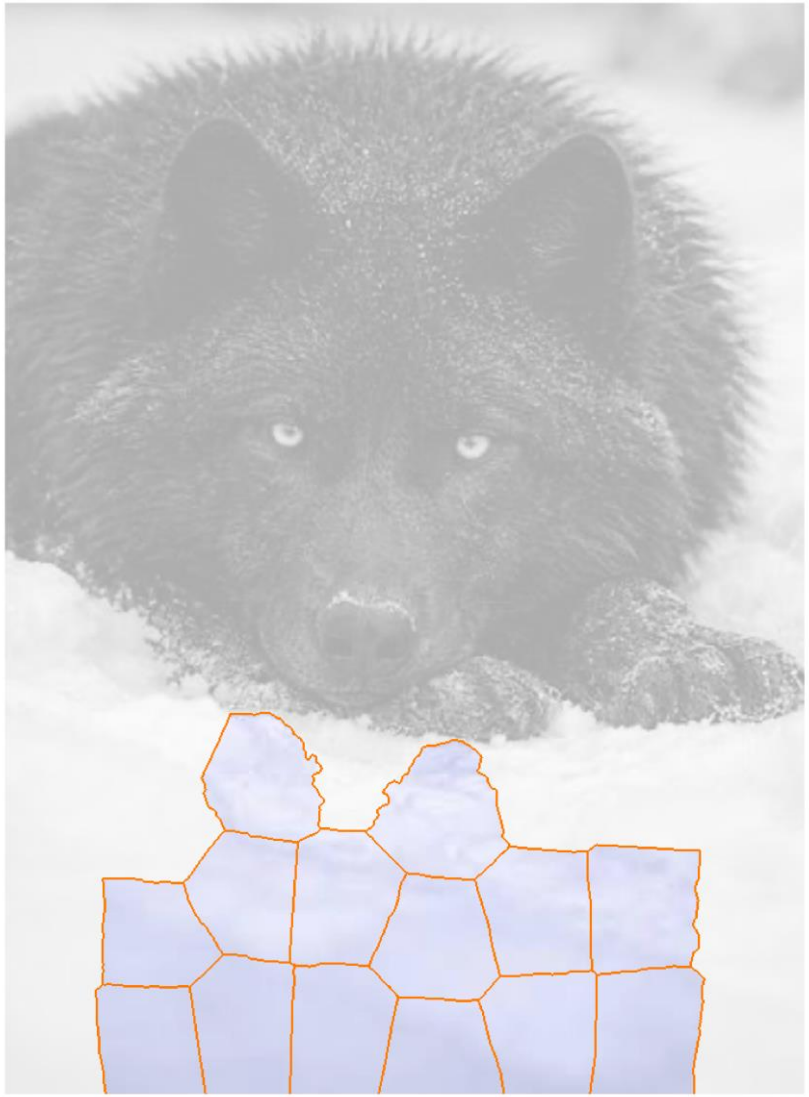
Starting with a trained model:

1. Divide the image into *superpixels*
2. Created perturbed images
3. Weight nearby samples (w_i)
4. **Locally fit a linear model**
5. Interpret the coefficients

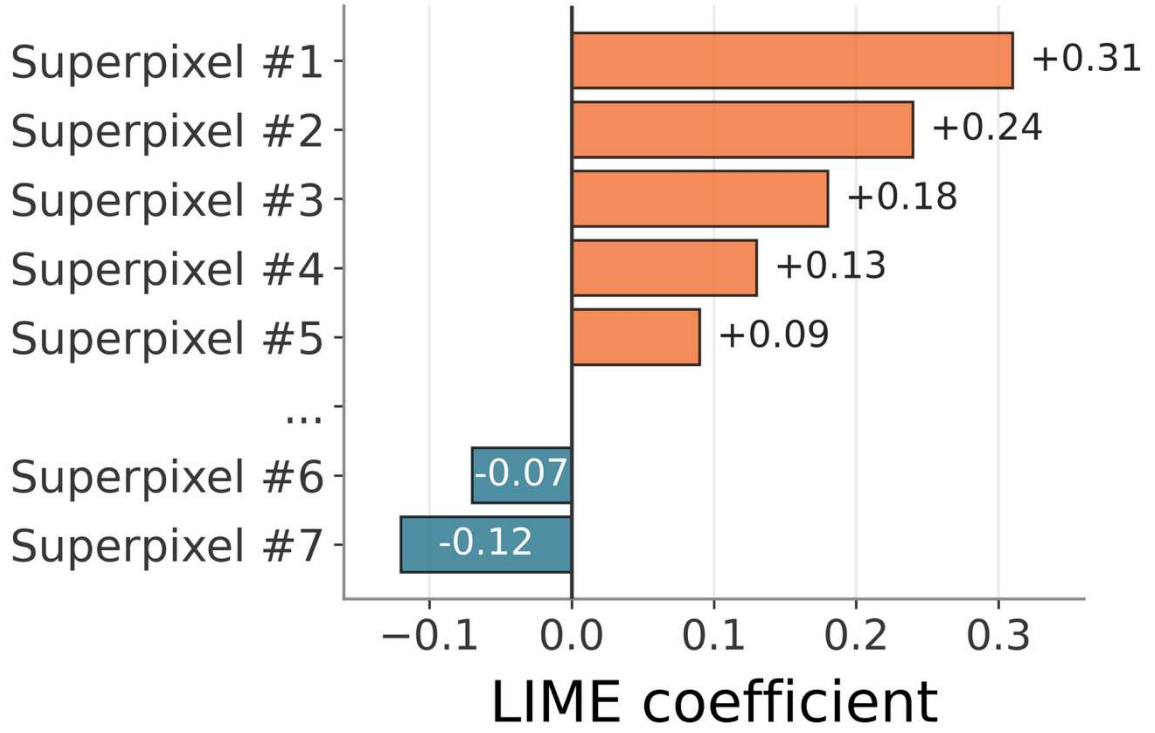
LIME for image classification: Results

We then typically visualize **which super pixels had the largest coefficients**

In this case, it is the snowy background that dictated the prediction



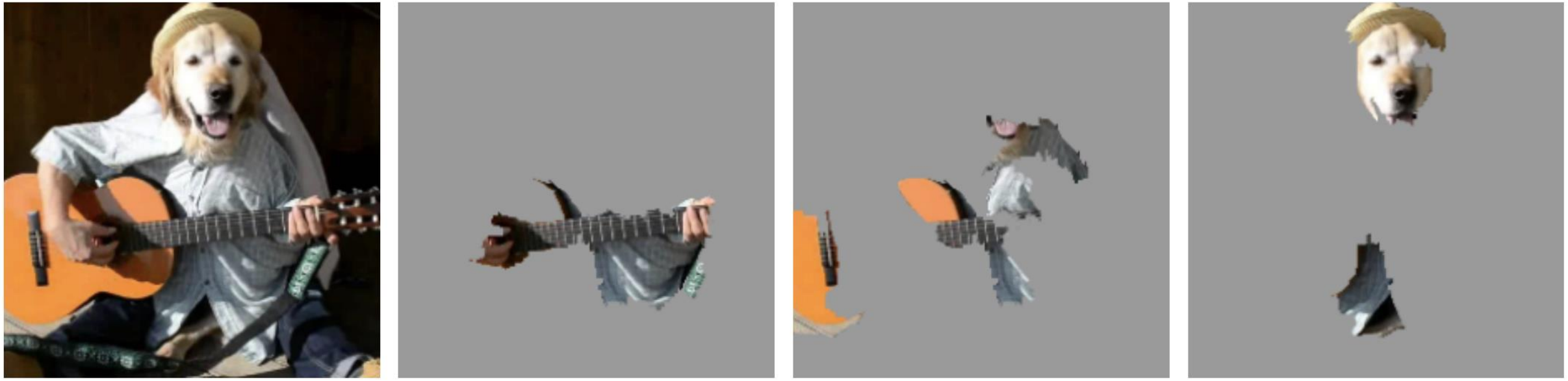
Each superpixel gets a coefficient



5. **Interpret the coefficients**

LIME for image classification: Results

From the original paper: we can also **break down feature importance by class**



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$), "Acoustic guitar" ($p = 0.24$) and "Labrador" ($p = 0.21$)

Questions?

Limitations of the LIME approach

LIME is useful but it has **drawbacks:**

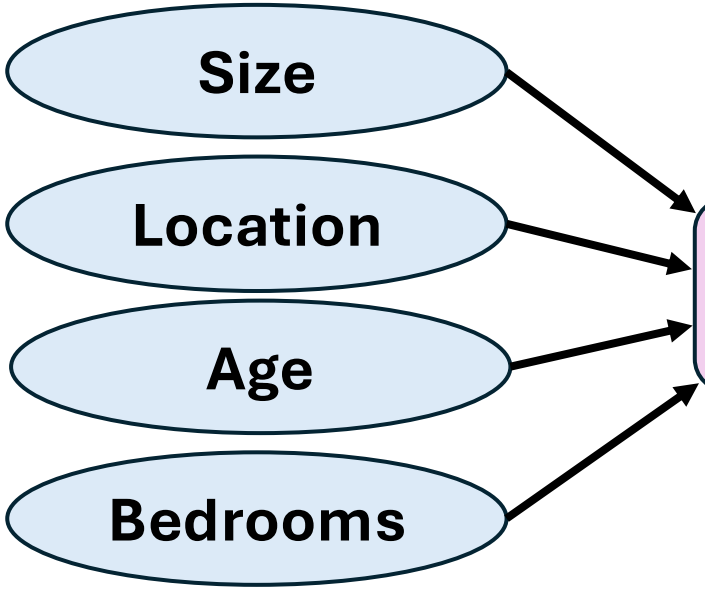
1. **It has some inherent random-ness (to generate perturbed samples)**
2. **It relies on user-chosen parameters (ϵ and σ) that can affect reliability**
3. **LIME coefficients may not be comparable across features with different units**

What we *want* is an explanation method that frames predictions as:

$$\textbf{Prediction} = \textbf{Baseline} + \textbf{Contributions from each feature}$$

$$f(x) = E[f(x)] + \phi_1 + \phi_2 + \cdots + \phi_p$$

SHAP: SHapley Additive exPlanations



SHAP asks: Which features are responsible for the added cost?

Model → **House price**

What portion of the $450k can we attribute to each feature?

Suppose our model predicts a house price of **$850k** on one sample



Baseline: Average prediction = **$400k** | **$450k contributions**

$$\textbf{Prediction} = \textbf{Baseline} + \textbf{Contributions from each feature}$$

$$f(x) = E[f(x)] + \phi_1 + \phi_2 + \cdots + \phi_p$$

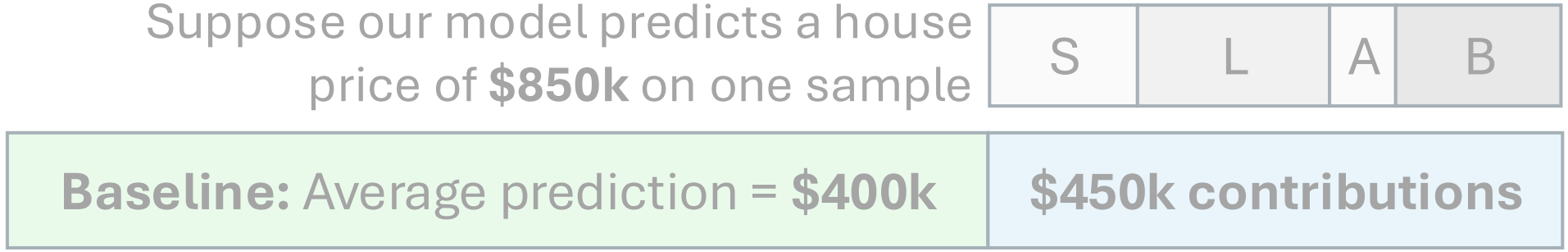
SHAP: SHapley Additive exPlanations

- Size contributes +$330k
- Location contributes +$170k
- Age contributes −$90k
- Bedrooms contributes +$40k

These values (ϕ_i) may be positive or negative, and they **should sum to the model prediction $f(x)$ minus the baseline**

How do we attribute credit fairly?

Suppose our model predicts a house price of **$850k** on one sample



$$\text{Prediction} = \text{Baseline} + \text{Contributions from each feature}$$

$$f(x) = E[f(x)] + \phi_1 + \phi_2 + \cdots + \phi_p$$

Shapley values in cooperative game theory

*Fair warning I know nothing about soccer/football

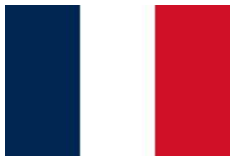


Argentina



vs.

France



Reward: $1,000,000

Suppose Argentina wins the game against France. How should we fairly **distribute the $1,000,000 amongst the four players?**

Shapley values = the portion of the reward that should be given to each player



From: Soledad Galli (https://www.youtube.com/watch?v=G2Qojy5lRtk)

Shapley values in cooperative game theory






Here we assume we can "simulate" the outcomes (we use our model)



Reward: $500,000
(they won half the time)

Suppose we take Messi out and then **Argentina wins only half the time**, meaning the avg reward is only half of one million: $500,000

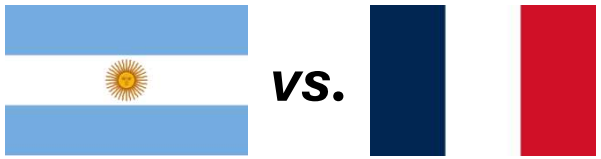
Does this mean that Messi should get paid $500,000? Because with him, they got the full $1,000,000?

From: Soledad Galli (https://www.youtube.com/watch?v=G2Qojy5lRtk)

Shapley values in cooperative game theory





Reward: $100,000

Suppose we now also take the goalie out (don't know his name) and then Argentina **wins only 10% of the time**, meaning the average reward is $100,000…

…But next, let's add Messi back in

From: Soledad Galli (https://www.youtube.com/watch?v=G2Qojy5lRtk)

Shapley values in cooperative game theory



Reward: $300,000

Adding Messi back in while keeping the goalie out raises the **win rate to 30%**

So based on this, Messi's contribution is only $200k, *not* the original $500,000 we thought (we went from $100k → $300k)

From: Soledad Galli (https://www.youtube.com/watch?v=G2Qojy5lRtk)

Shapley values in cooperative game theory

To get Messi's fair contribution, let's **average over all possible groups with and without Messi**



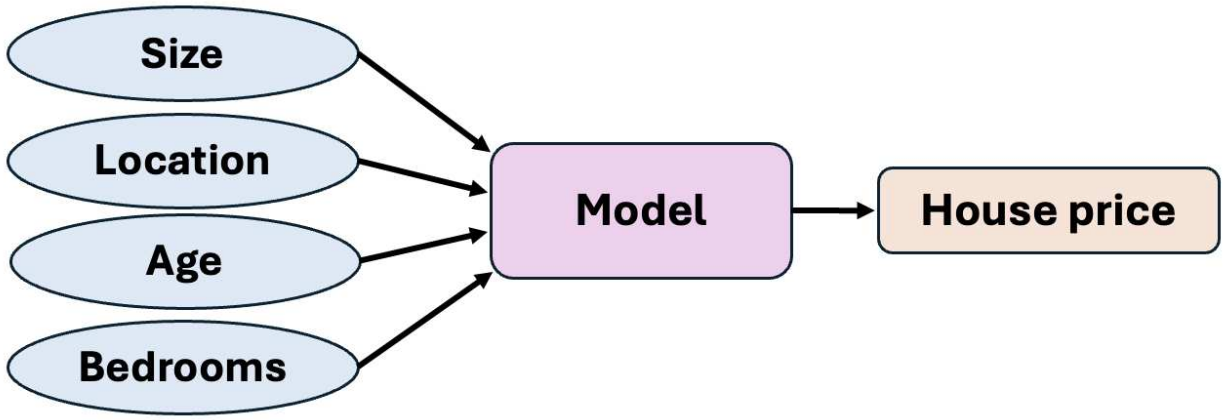
Example:

With nobody playing, win rate = 0%

With just Messi playing, win rate = 5%

The Shapley value for Messi is given by his average contribution over these groups; **how does the reward change when he is added, on average?**

SHAP: Step-by-step

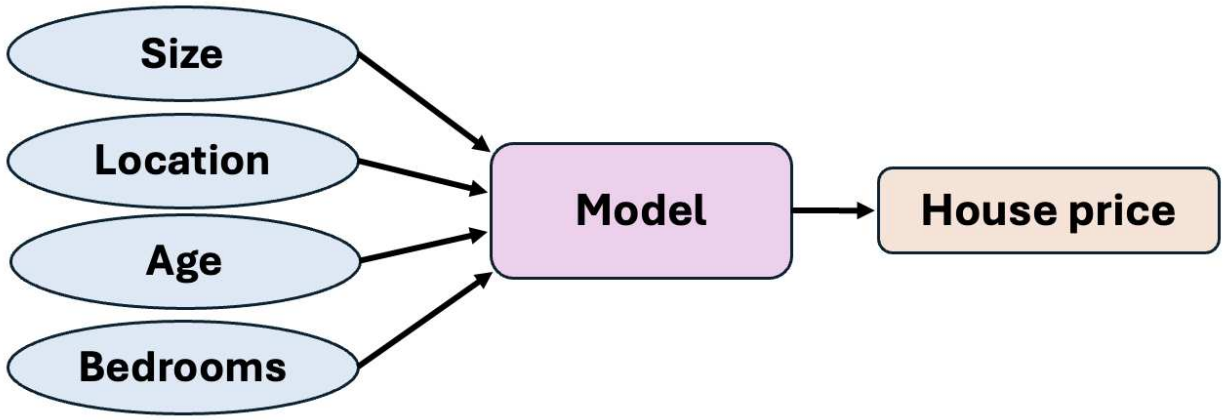


Suppose our training dataset contains N houses. Compute the baseline prediction by **averaging the predicted prices of all N houses:**

$$E[f(x)] = \frac{1}{N}\sum_{i=0}^{N} f(x_n)$$

1. **Compute the baseline prediction**
2. Choose one sample to inspect
3. Enumerate subsets of features
4. Compute values for each subset
5. Add a feature and recompute
6. Average over all contributions

SHAP: Step-by-step



$$(x_1, x_2, x_3, x_4) = (\text{size}, \text{location}, \text{age}, \text{bed})$$

$$= (2500\ \text{ft}^2, 2\ \text{miles}^*, 10\ \text{yrs}, 2\ \text{bed})$$

Just one sample, one prediction

Model $f(x)$

$850,000

What portion of this cost, relative to $400k baseline, comes from each feature?

*From downtown LA

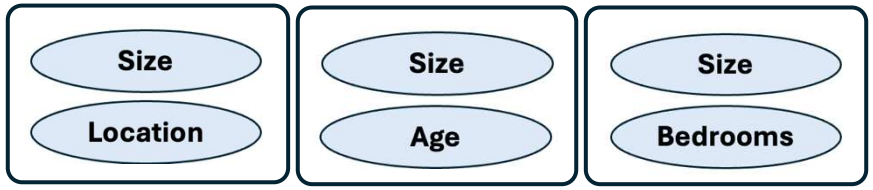
1. Compute the baseline prediction
2. **Choose one sample to inspect**
3. Enumerate subsets of features
4. Compute values for each subset
5. Add a feature and recompute
6. Average over all contributions

SHAP: Step-by-step

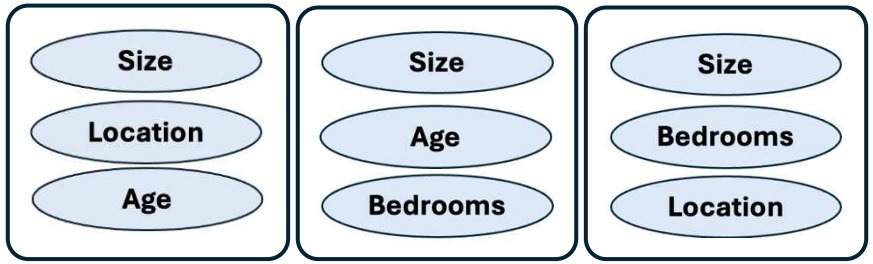
For p predictors there are $\mathbf{2^p}$ **subsets**



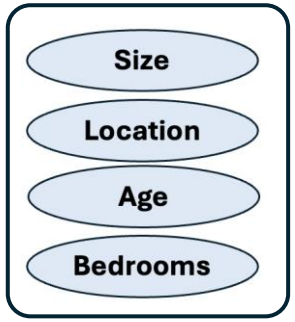
Null set $\emptyset$



$\{x_1, x_2\}$
$\{x_1, x_3\}$, …



$\{x_1, x_2, x_3\}$
$\{x_1, x_3, x_4\}$,
…



Full set:
$\{x_1, x_2, x_3, x_4\}$

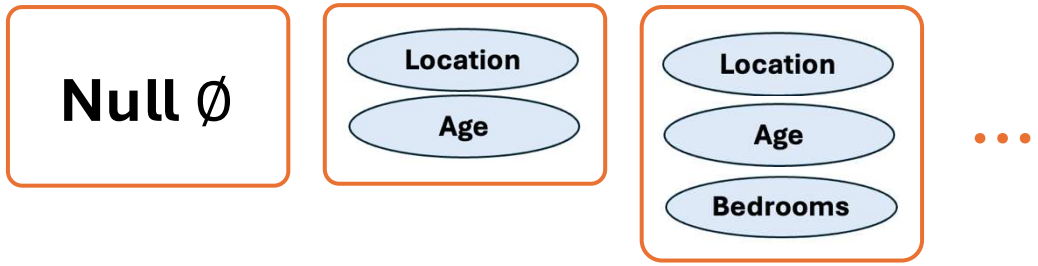
Here we have 4 predictors, 16 subsets

1. Compute the baseline prediction
2. Choose one sample to inspect
3. **Enumerate subsets of features**
4. Compute values for each subset
5. Add a feature and recompute
6. Average over all contributions

SHAP: Step-by-step

To calculate the **contribution** of **house size** to its price:

a. Consider all **subsets *without* size**



b. **Make predictions of sales price using each subset**

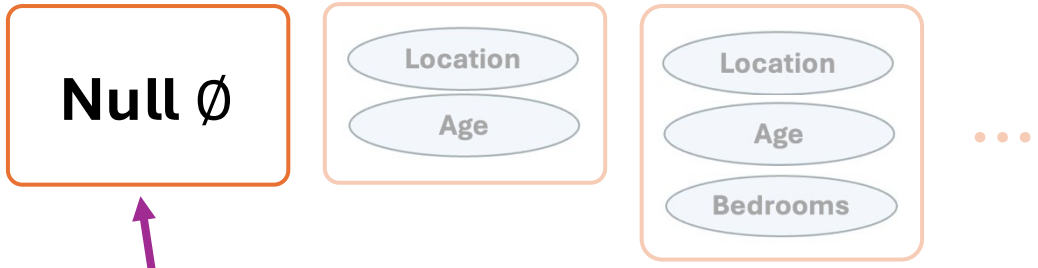
But how do we make predictions with certain features held out?

1. Compute the baseline prediction
2. Choose one sample to inspect
3. Enumerate subsets of features
4. **Compute values for each subset**
5. Add a feature and recompute
6. Average over all contributions

SHAP: Step-by-step

To calculate the **contribution** of **house size** to its price:

a. Consider all **subsets *without* size**



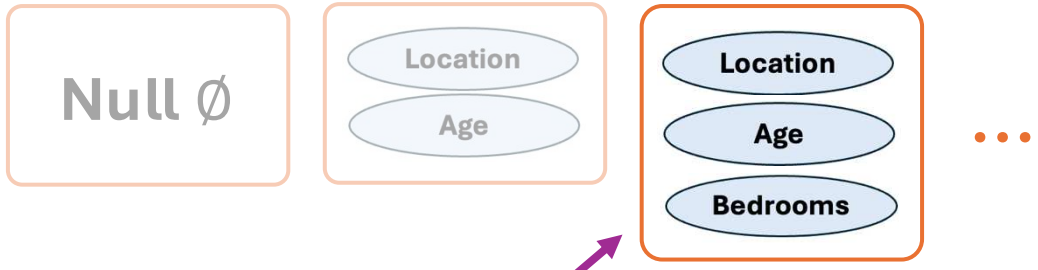
$f(\emptyset)$ = the **average** sales price across the dataset

1. Compute the baseline prediction
2. Choose one sample to inspect
3. Enumerate subsets of features
4. **Compute values for each subset**
5. Add a feature and recompute
6. Average over all contributions

SHAP: Step-by-step

To calculate the **contribution** of **house size** to its price:

a. Consider all **subsets *without* size**



$$f(x_{i\neq \text{size}}) = \frac{1}{N}\sum_{n=1}^{N} f(x_{n,\text{size}}, x_{\text{others}})$$

Enumerate possible values from the ***entire dataset***

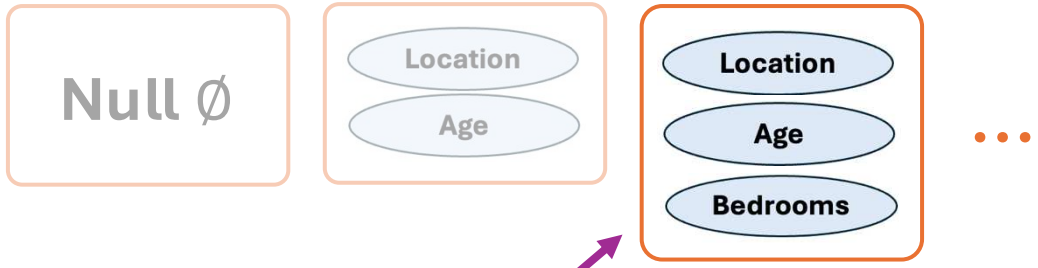
Fixed values ***for our chosen sample***

1. Compute the baseline prediction
2. Choose one sample to inspect
3. Enumerate subsets of features
4. **Compute values for each subset**
5. Add a feature and recompute
6. Average over all contributions

SHAP: Step-by-step

To calculate the **contribution** of **house size** to its price:

a. Consider all **subsets *without* size**



$$f(x_{i \neq \text{size}}) = \frac{1}{N}\sum_{n=1}^{N} f(x_{n,\text{size}}, x_{\text{others}})$$

Enumerate possible values from the ***entire dataset***

Fixed values ***for our chosen sample***

Recall our chosen sample:

$$(x_1, x_2, x_3, x_4) = (\text{size}, \text{location}, \text{age}, \text{bed})$$
$$= (2500\ \text{ft}^2, 2\ \text{miles}^*, 10\ \text{yrs}, 2\ \text{bed})$$

Model $f(x)$

$850,000

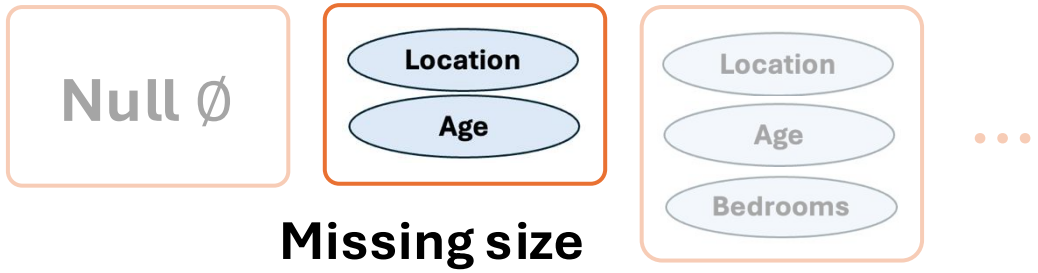
- **Location, age, and bed are fixed**
- But now we "remove" the specific size, instead enumerating over all house sizes in our training dataset

$$f(1200\ \text{ft}^2, 2\ \text{miles}, 10\ \text{yrs}, 2\ \text{bed}) + f(1600\ \text{ft}^2, 2\ \text{miles}, 10\ \text{yrs}, 2\ \text{bed}) + \cdots$$

SHAP: Step-by-step

To calculate the **contribution** of **house size** to its price:

a. Consider all **subsets *without* size**



Missing size *and* bedrooms

$$f(1200\ \text{ft}^2, 2\ \text{miles}, 10\ \text{yrs}, 1\ \text{bed}) + f(1600\ \text{ft}^2, 2\ \text{miles}, 10\ \text{yrs}, 3\ \text{bed}) + \cdots$$

We can write a similar equation for *several* missing features

Recall our chosen sample:

$$(x_1, x_2, x_3, x_4) = (\text{size}, \text{location}, \text{age}, \text{bed})$$
$$= (2500\ \text{ft}^2, 2\ \text{miles}^*, 10\ \text{yrs}, 2\ \text{bed})$$

Model $f(x)$

$850,000

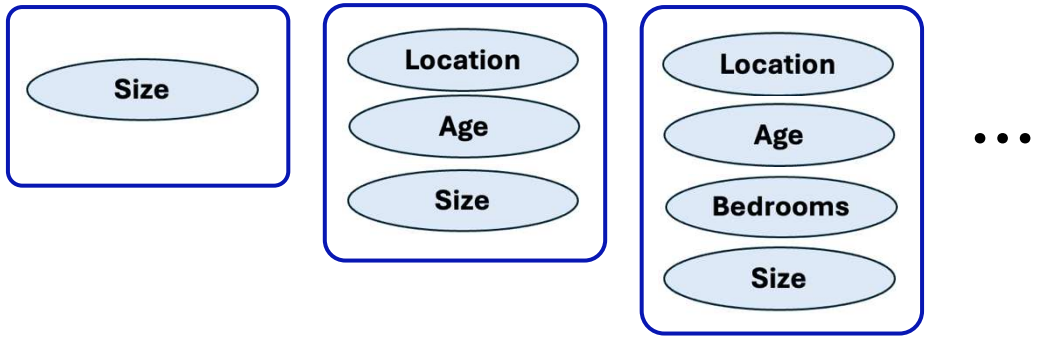
- **Location, age, and bed are fixed**
- But now we "remove" the specific size, instead enumerating over all house sizes in our training dataset

$$f(1200\ \text{ft}^2, 2\ \text{miles}, 10\ \text{yrs}, 2\ \text{bed}) + f(1600\ \text{ft}^2, 2\ \text{miles}, 10\ \text{yrs}, 2\ \text{bed}) + \cdots$$

SHAP: Step-by-step

To calculate the **contribution** of **house size** to its price:

c. Then **add size** to those subsets

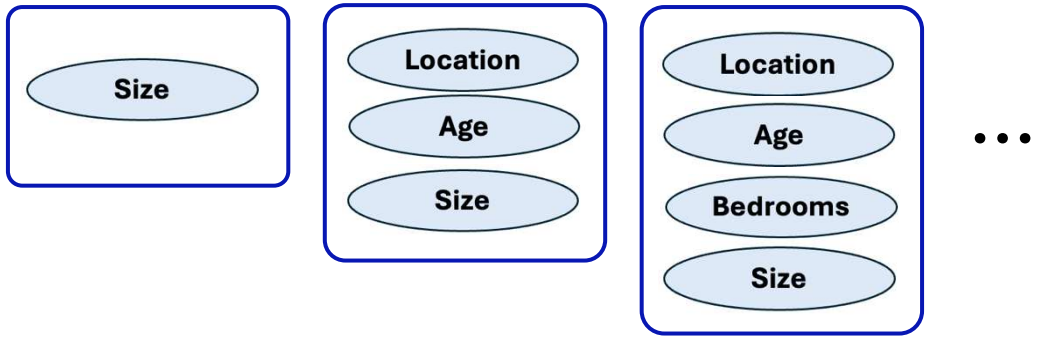


1. Compute the baseline prediction
2. Choose one sample to inspect
3. Enumerate subsets of features
4. Compute values for each subset
5. **Add a feature and recompute**
6. Average over all contributions

SHAP: Step-by-step

To calculate the **contribution** of **house size** to its price:

c. Then **add size** to those subsets



...

d. Compute $f(x, \text{with size})$ for each

$$= \frac{1}{N} \sum_{n=1}^{N} f(x_{n,\text{missing}}, x_{\text{included}})$$

Now size will be part of the included features

1. Compute the baseline prediction
2. Choose one sample to inspect
3. Enumerate subsets of features
4. Compute values for each subset
5. **Add a feature and recompute**
6. Average over all contributions

SHAP: Step-by-step

To calculate the **contribution** of **house size** to its price:

e. Compute the **difference** between the function values ***with* and *without* size**, for every set

$$\Delta f = f(x, \text{with size}) - f(x, \text{without size})$$

1. Compute the baseline prediction
2. Choose one sample to inspect
3. Enumerate subsets of features
4. Compute values for each subset
5. Add a feature and recompute
6. **Average over all contributions**

SHAP: Step-by-step

To calculate the **contribution** of **house size** to its price:

e. Compute the **difference** between the function values **with and without size**, for every set

$$\Delta f = f(x, \text{with size}) - f(x, \text{without size})$$

f. Average over all those differences

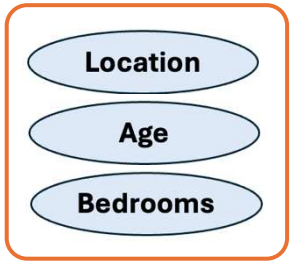
$$\sum_j w_j \Delta f_j$$

Where w_j accounts for **permutations**

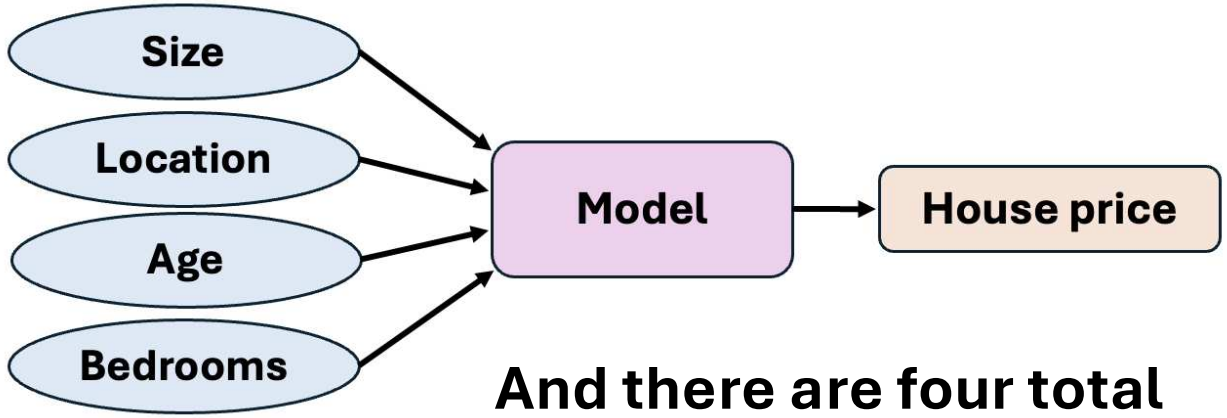
$$w_j = \frac{|\text{Set } j|!\,(p - |\text{Set } j| - 1)!}{p!}$$

1. Compute the baseline prediction
2. Choose one sample to inspect
3. Enumerate subsets of features
4. Compute values for each subset
5. Add a feature and recompute
6. **Average over all contributions**

Permutations in SHAP



For this set, size = 3



And there are four total predictors ($p = 4$)

The size of set j (how many features it includes)

The total number of features

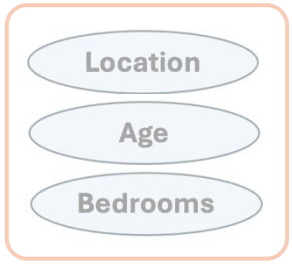
$$w_j = \frac{|\text{Set } j|!\,(p - |\text{Set } j| - 1)!}{p!}$$

$$w_j = \frac{(3)!\,(4 - 3 - 1)!}{4!}$$

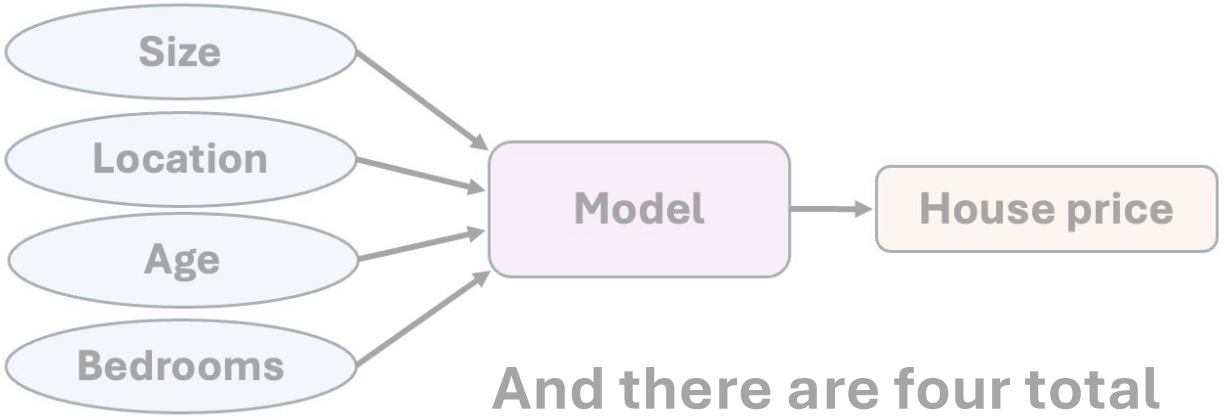
$$= \frac{(3)(2)(1)(0)!}{(4)(3)(2)(1)} = \frac{6}{24}$$

What does this mean?

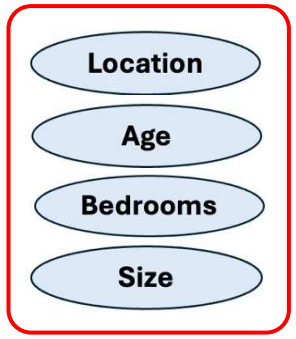
Permutations in SHAP



For this set,
size = 3



And there are four total predictors ($p = 4$)



- There are 24 possible permutations of this set.
- **In 6 of these:** location, age, and bedrooms come *before* house size

$$w_j = \frac{(3)!\,(4-3-1)!}{4!}$$

$$= \frac{(3)(2)(1)(0)!}{(4)(3)(2)(1)} = \frac{6}{24}$$

What does this mean?

Why order matters

Suppose we are evaluating **house size** and **location**

With house size only, model predicts $200k sales price

With location only, model also predicts $200k sales price

With both features, model predicts $400k sales price

Which features gets the credit will depend on the order:

- **“I added house size and price went up $200k!”**
- **“I added location and price went up $200k!”**

Neither is more current than the other – so we should average over them

Questions?

SHAP in one equation

SHAP value of feature j

Weighting to account for different orderings

$$\phi_j = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} [f(S \cup \{j\}) - f(S)]$$

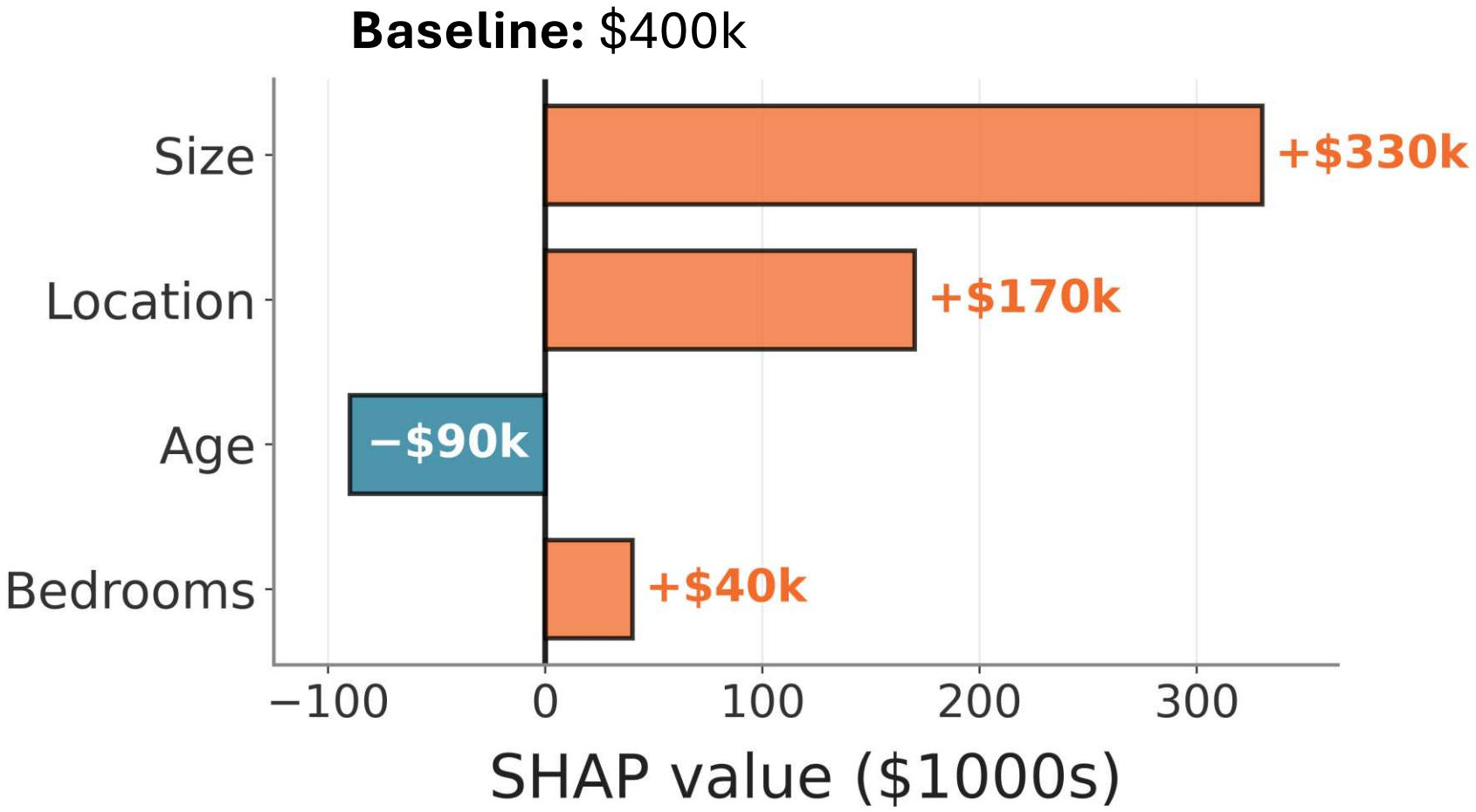
Sum over all differences: predictions with and without feature j

- $F \setminus \{j\}$= set of all features that don't have j
- j = feature whose SHAP value we're computing
- S = any subset of the other features, excluding j
- $|S|$ = number of features in that subset

Compute ϕ_j for all features and plot the results

*There will be concrete examples in discussion, and in my final review

Plotting SHAP values

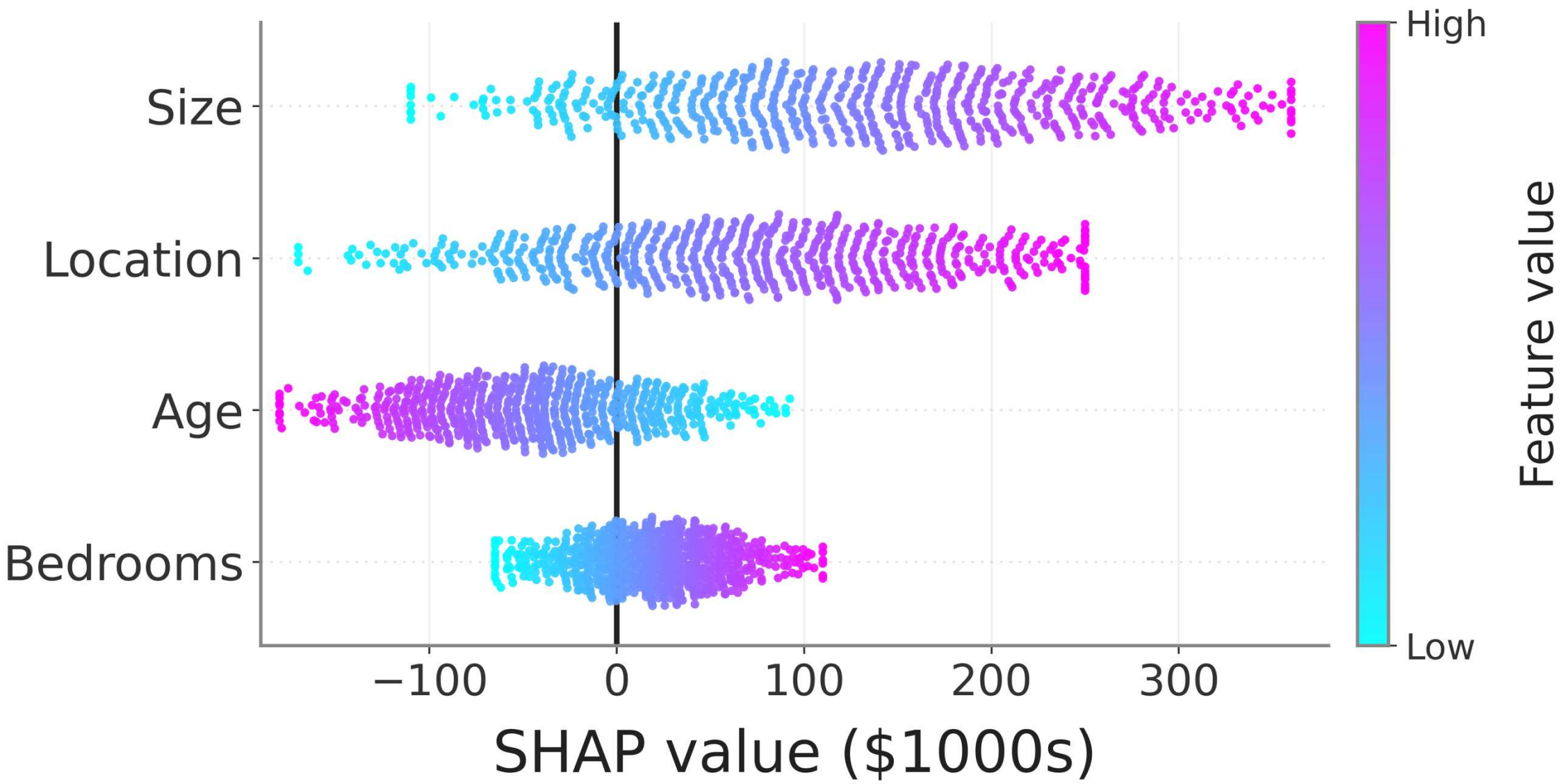


Baseline: $400k	**$450k contributions**

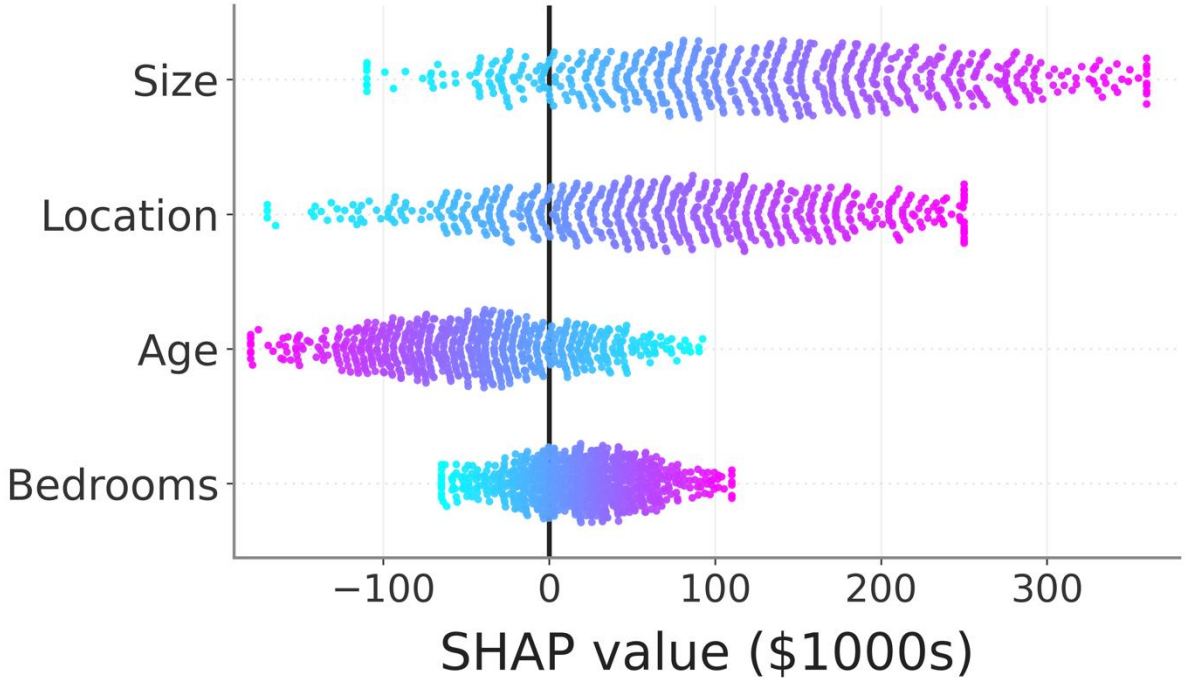
Remember, this is ***local*** – i.e., for one sample. But people will often plot global ***distributions*** of SHAP values across all samples…

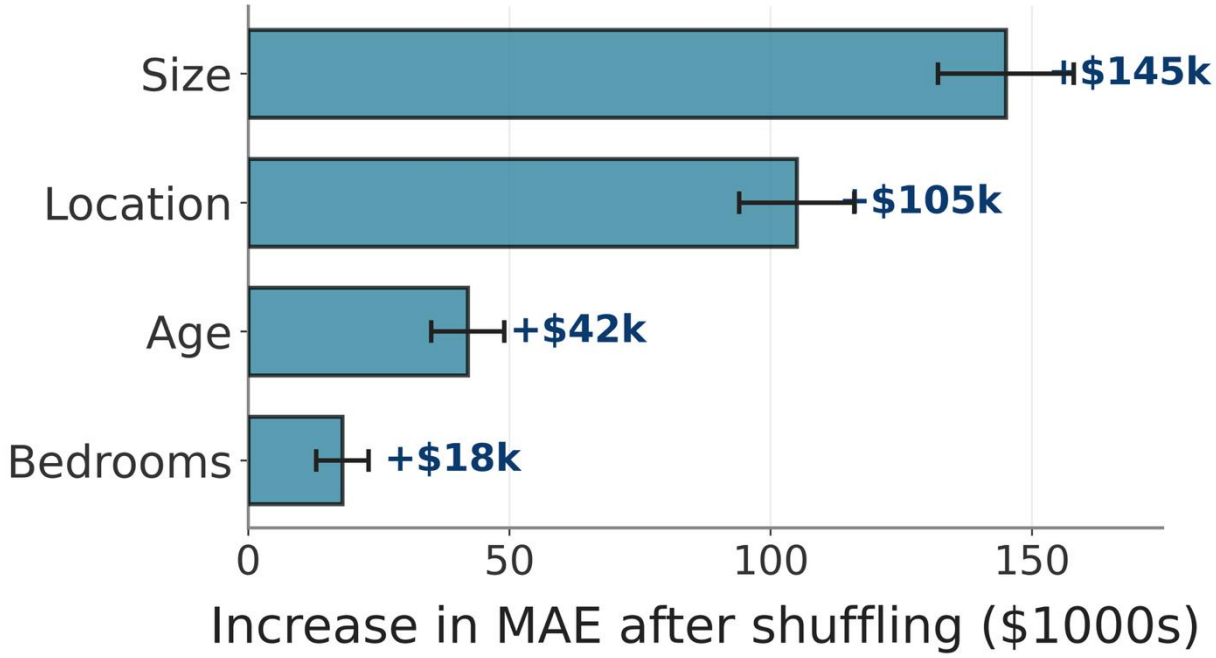
As a sanity check, **make sure the SHAP vales add up to the total expected contributions**

Plotting *all* SHAP values in a “beeswarm” plot



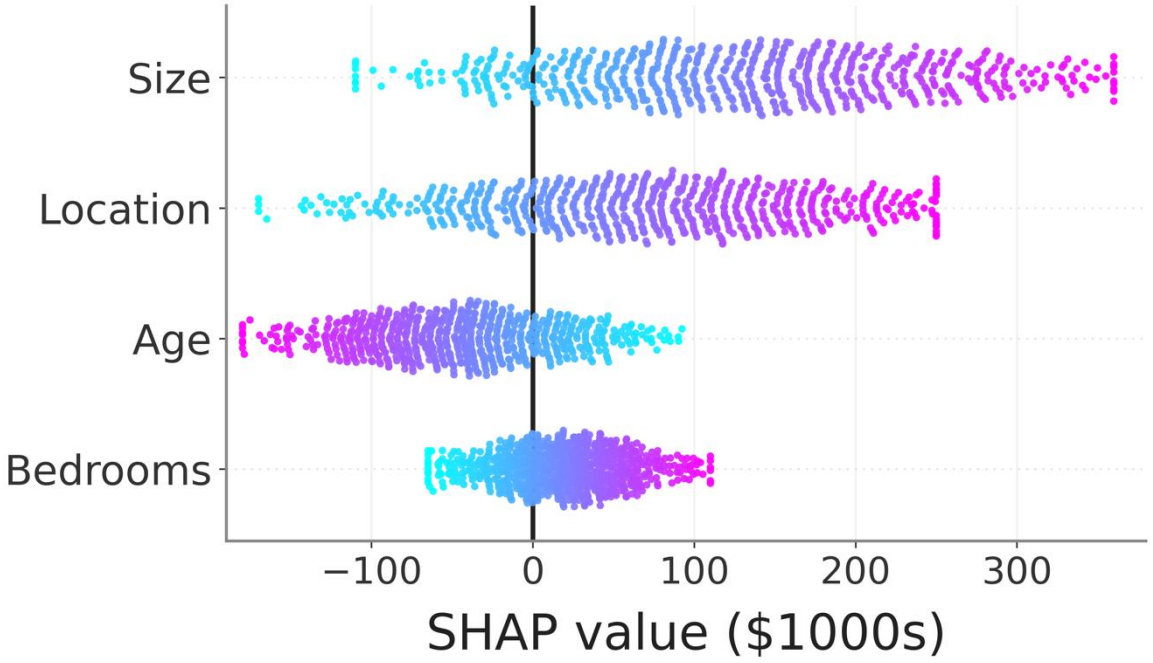
SHAP *vs.* permutation importance

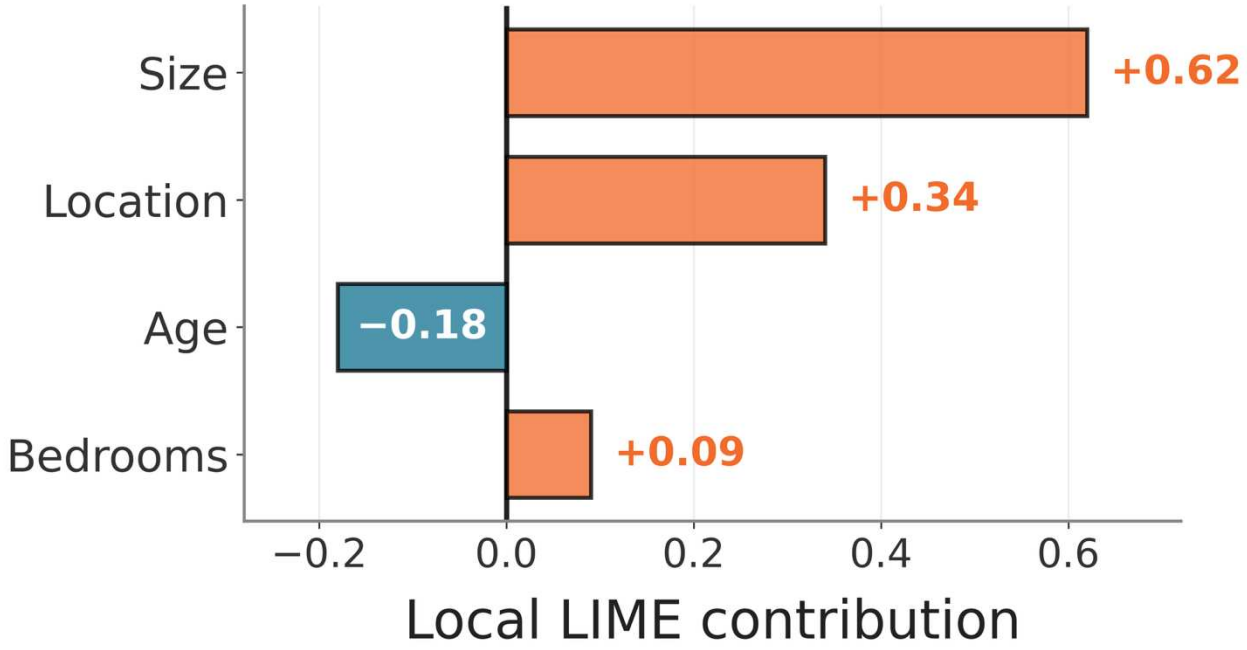




- This may remind you of earlier permutation importance plots…But remember: **permutation importance is global whereas SHAP values are local**
- **SHAP values also give information regarding *directionality* –** in other words, not only how much each feature affects the output, but in which direction **(+/-)**

SHAP *vs.* LIME





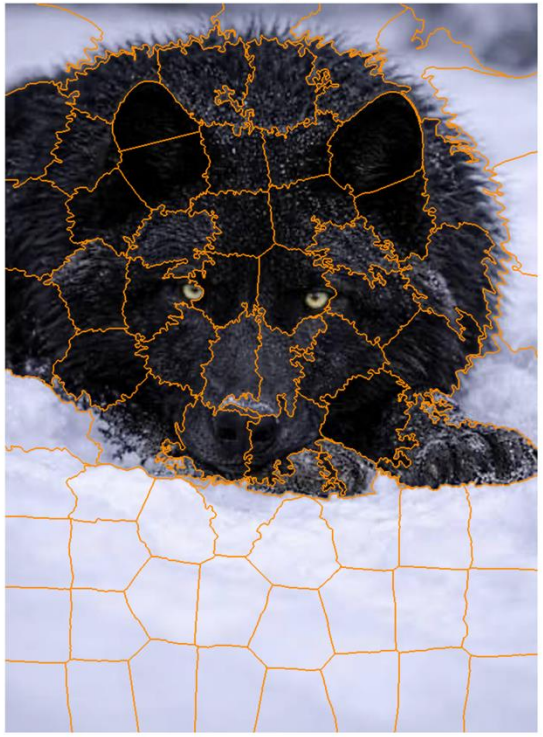
- Notice that the LIME contributions are in raw units and don't sum to $450k… **LIME values are in general less directly interpretable than SHAP**
- Additionally, LIME coefficients vary a lot from sample to sample (different fits), and as such it is less common to make beeswarm plots for LIME coefficients

SHAP for images: Same ideas apply!

Original image



Superpixels

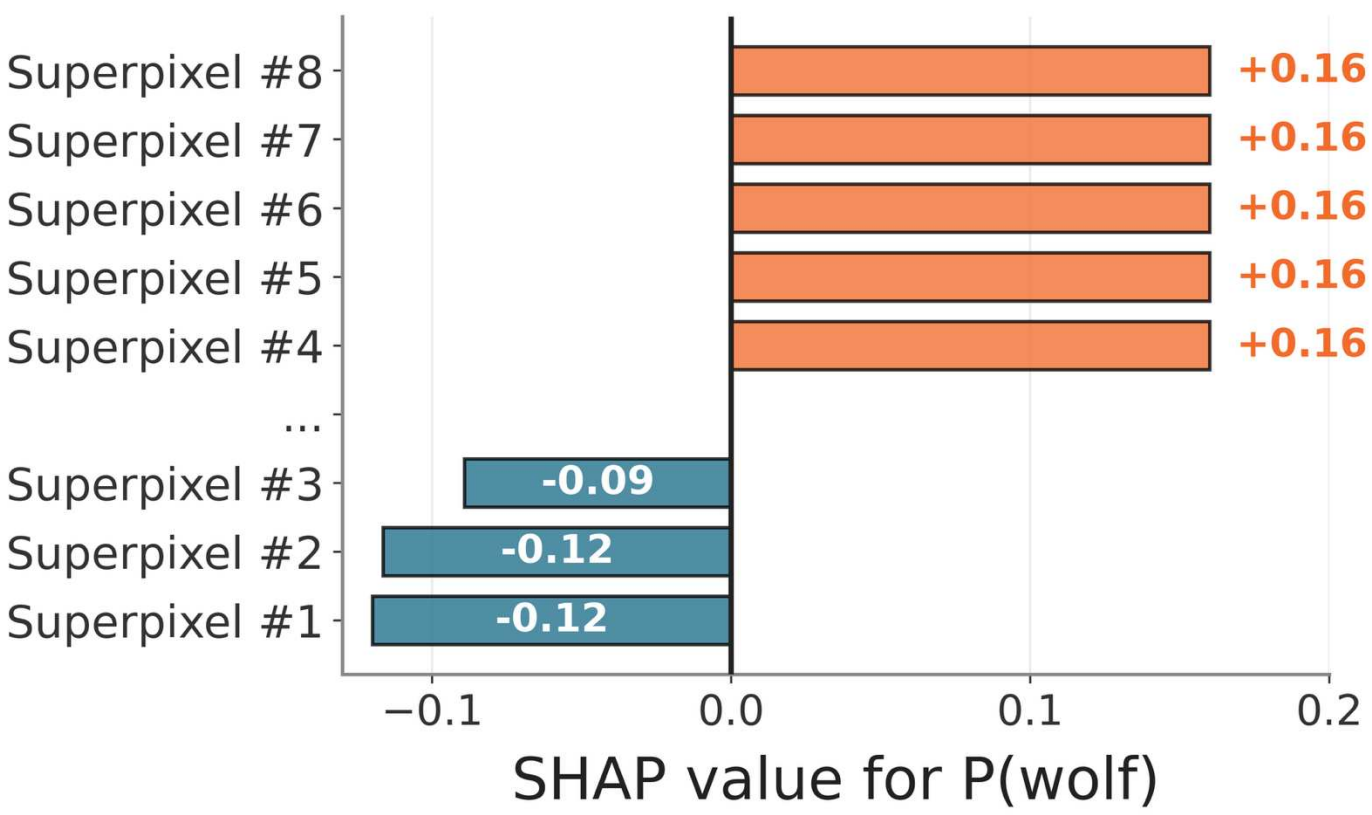


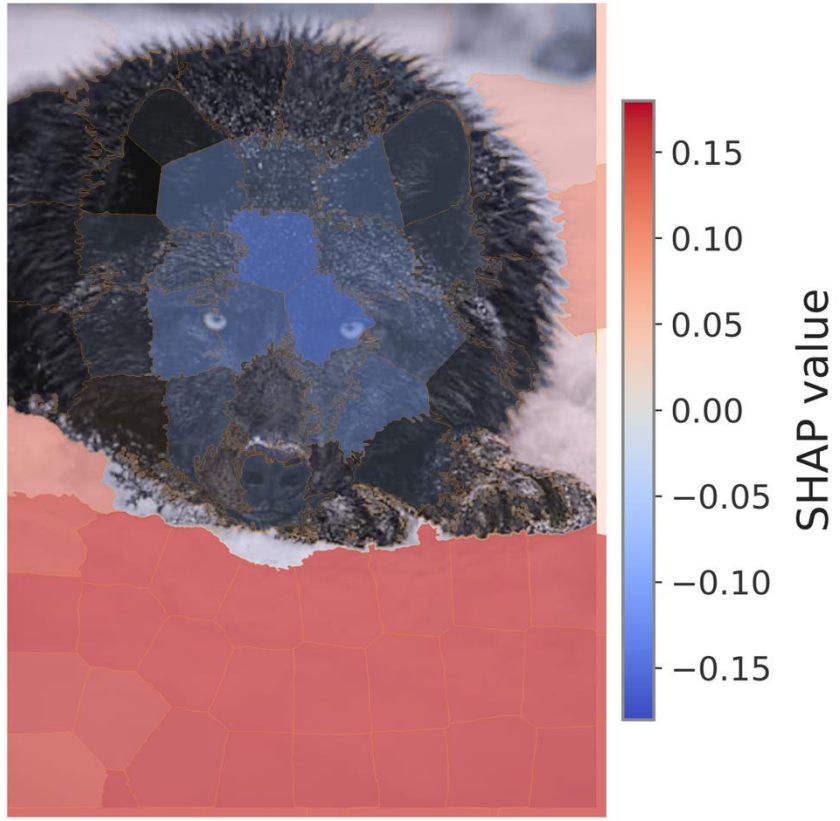
Now each feature is a superpixel

"Without a feature" – black out that part of the image, or average its color based on the rest of the image

1. Compute baseline ***over all images***
2. Choose one ***image*** to inspect
3. Create ***superpixels of the image***
4. Predict on sets ***without a superpixel***
5. Add ***that superpixel*** and recompute
6. Average over all contributions

SHAP for images: Same ideas apply!





SHAP treats each superpixel as a feature and assigns it a contribution.

Positive values increase the wolf prediction relative to the baseline;

negative values decrease it.

Questions?

What to know for the final exam

- Distinguish **global *vs.* local explanations**
- **Understand SHAP and LIME, interpret their coefficients**
- **How to do a SHAP calculation**, and why order matters