Uncertainty and Interpretability in Machine Learning Models

Joshua S. Speagle (沈佳士)

Department of Statistical Sciences
David A. Dunlap Department of Astronomy & Astrophysics
Dunlap Institute for Astronomy & Astrophysics
Data Sciences Institute

UNIVERSITY OF TORONTO

Overview: Part 1 (Uncertainty)

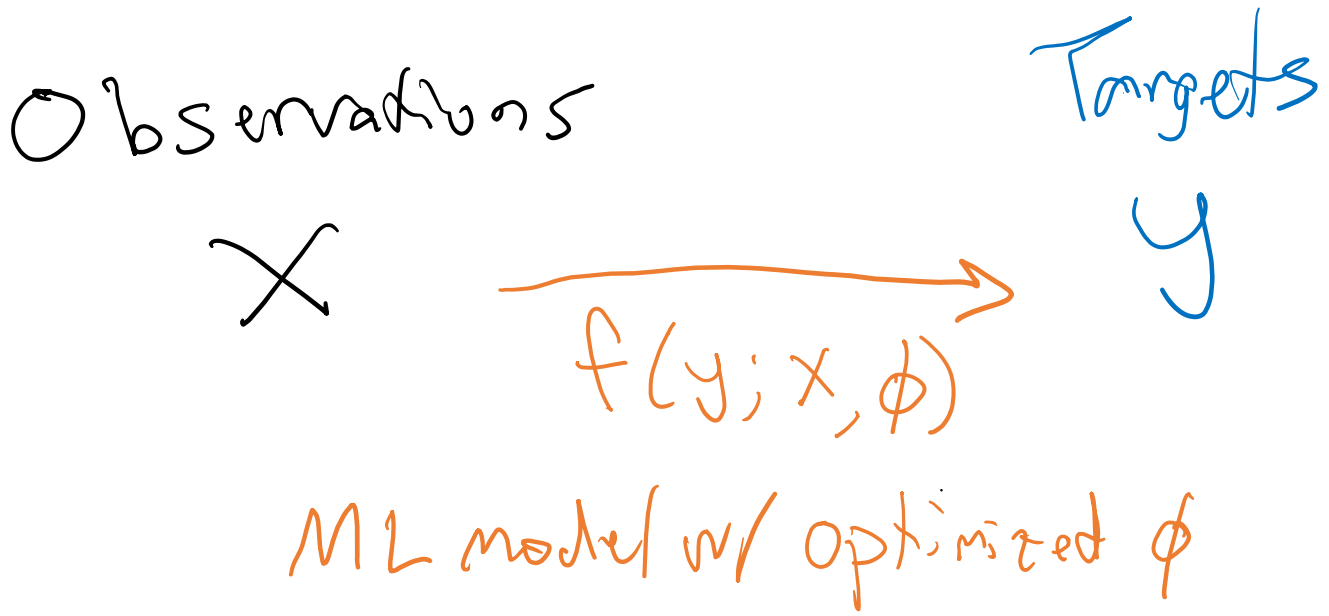
- **Uncertainties are ubiquitous in the physical sciences** and come in many different forms.

- These **will** have an impact on any downstream machine learning-oriented application.

- Questions:
  - **How much** do we have to worry about this in theory and practice?
  - How do **existing methods** implicitly or explicitly deal with these?
  - Are there easy **strategies** to incorporate errors into existing methods if they don't?

Opening Discussion Question

- Consider the classic ML workflow sketched out below. How might uncertainties affect this workflow?
  - If it helps, consider a motivating example from your own work.



Types of Uncertainties

- Inputs
  - Observables (x)
  - Targets (y)
- Model
  - Hyperparameters (ϕ)
  - Predictions (s_y)

Motivating example:
Dust mapping

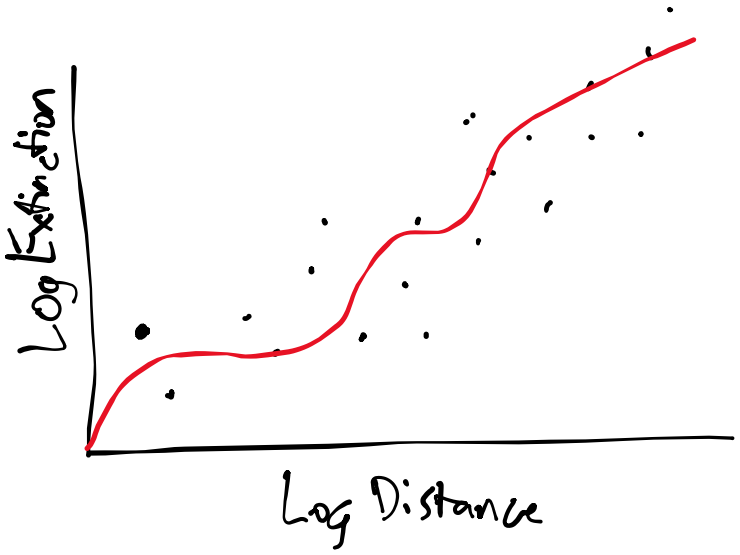
Green et al. (2019)

Idea 1: Just add the errors in? …Maybe?

- Does this make sense?

- Does it work?

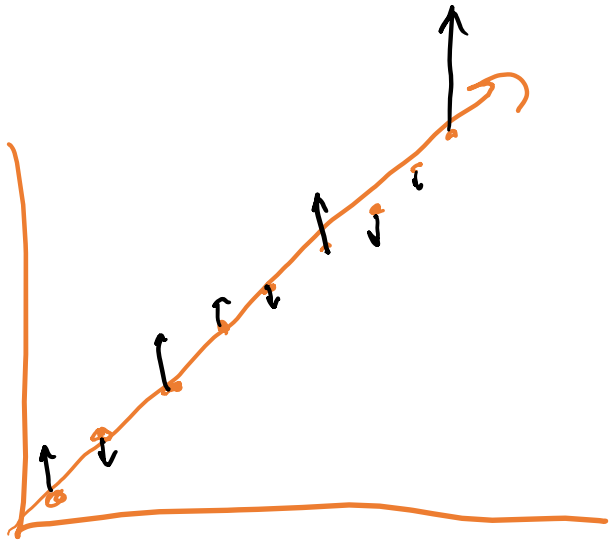
Motivating example:
Dust mapping



Back to Basics: Linear Regression

- To understand what errors do, let's go back to basics.

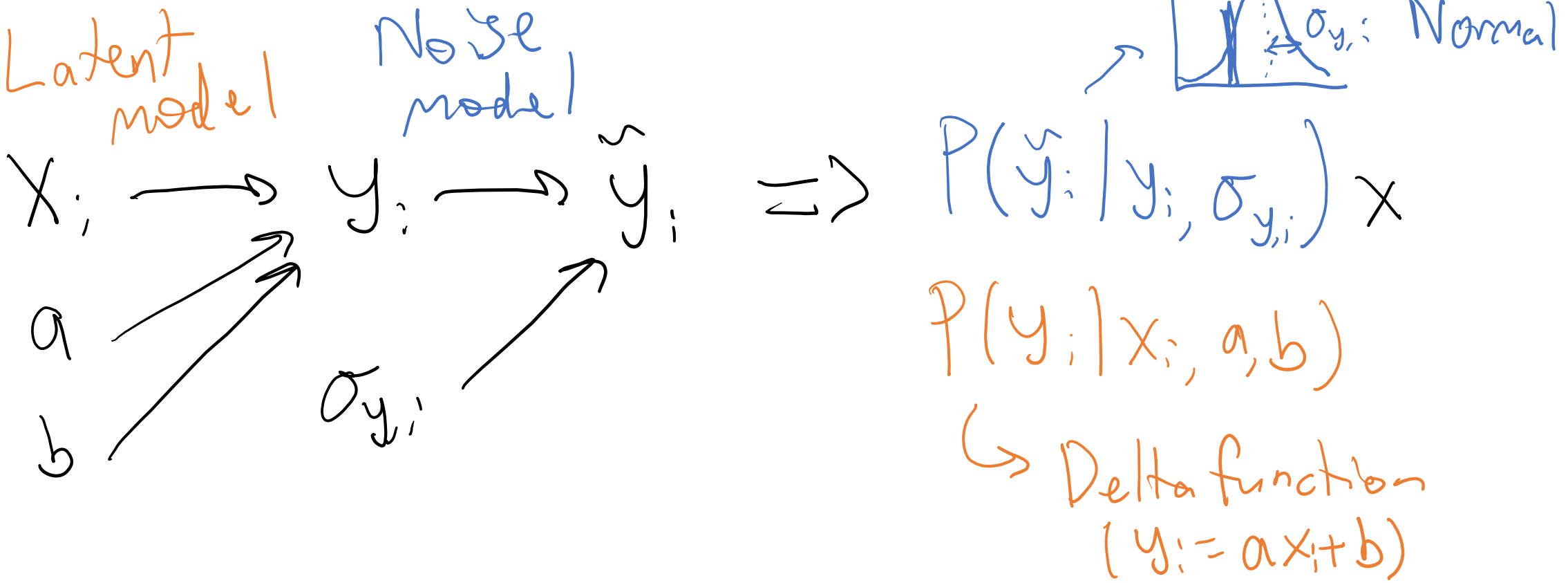
$i = 1, \dots, n: X_i \overset{i.i.d}{\sim} P(x)$ → $y_i = a x_i + b$



$\tilde{y} = y_i + \epsilon_i, \quad \epsilon_i \overset{iid}{\sim} N(0, \sigma_{y,i})$

Building Basic Probabilistic Models

- We can now construct a dependency graph showing what happens here and turn it into a probability distribution.



Dealing with Latent Parameters

- What do we do about our latent (unobserved) parameters?

$$P(\tilde{y}_i \mid \sigma_{y,i}, x_i, a, b) = \int P(\tilde{y}_i \mid y_i, \sigma_{y,i}) P(y_i \mid x_i, a, b)\, dy_i$$

- Marginalizing either can be done **analytically** (integrate terms out) or **numerically** (sample for latent parameters)

For $y_i = ax + b_i \longrightarrow P(\tilde{y}_i \mid ax_i + b, \sigma_{y,i})$

Lessons Learned: Linear Regression

- We can now show:

If $\sigma_{y,i} = \sigma_y = \text{constant}$:

$$\text{Loss} = -2 \ln L = \text{MSE}$$

Otherwise:

$$\text{Loss} = \text{Mean-}\chi^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{y}_i - y_{A,i}}{\sigma_{y,i}}\right)^2$$

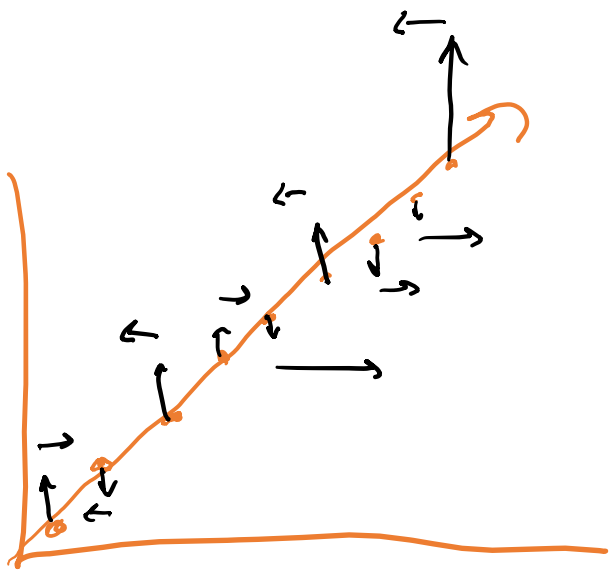
- **Takeaway #1:** There is a **unique loss function** associated with the (negative log-)probability given your noise model.

- **Takeaway #2:** Inference with noisy data requires **marginalizing** over some unknown noiseless (latent) values.

From Output to Input Errors

- What about errors on the **input** values?

$i = 1, \dots, n: X_i \overset{iid}{\sim} P(x)$



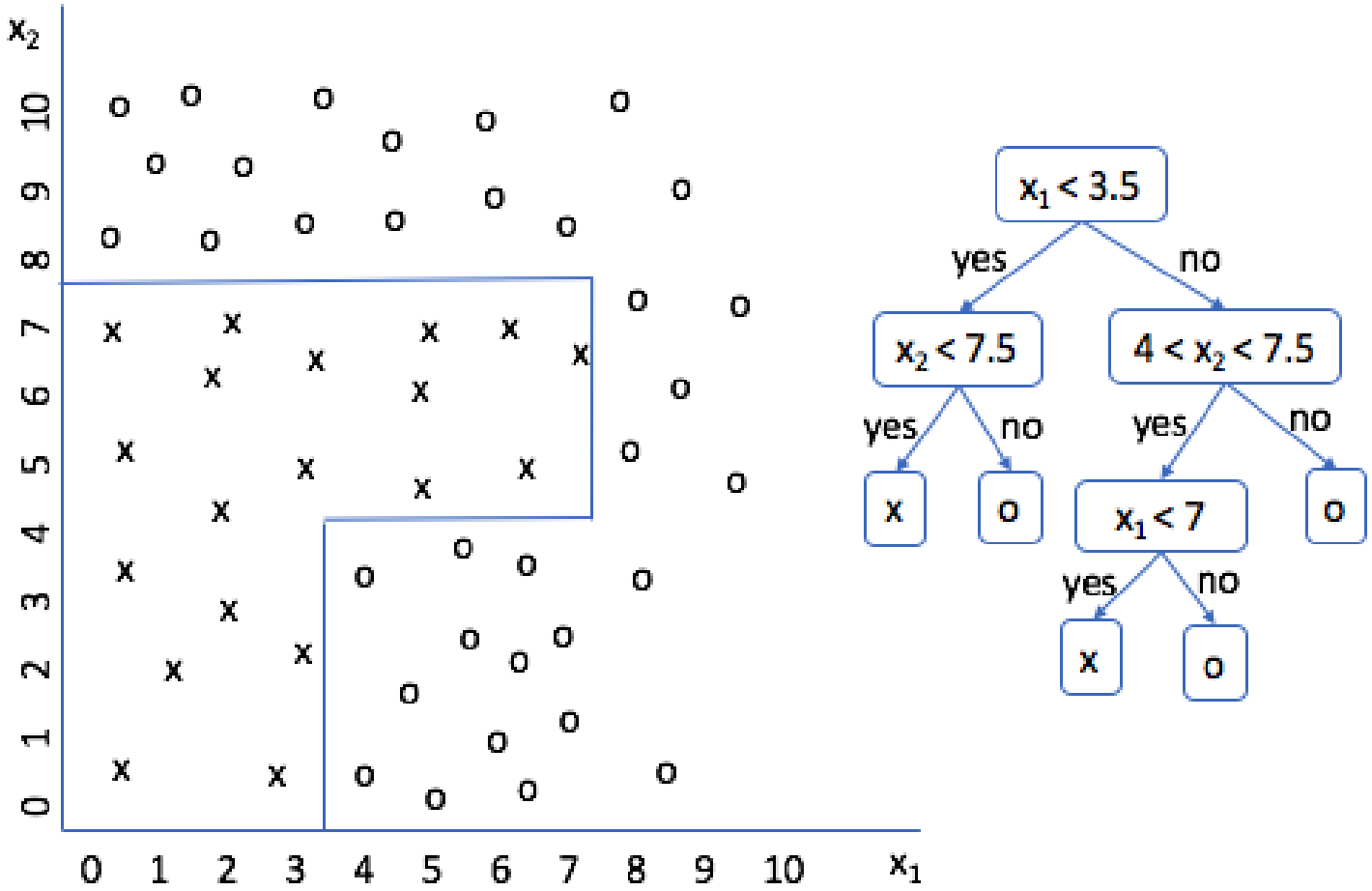
$y_i = a x_i + b$

$\tilde{x}_i = x_i + \delta_i, \quad \delta_i \overset{iid}{\sim} N(0, \sigma_{x,i})$

$\tilde{y} = y_i + \epsilon_i, \quad \epsilon_i \overset{iid}{\sim} N(0, \sigma_{y,i})$

Idea 2: Monte Carlo it?

- Assume you have a trained **decision tree** model where you have ignored the errors. You propose incorporating the uncertainties by querying the tree using **Monte Carlo samples** taken from the noise distribution. Will this work? Why or why not?

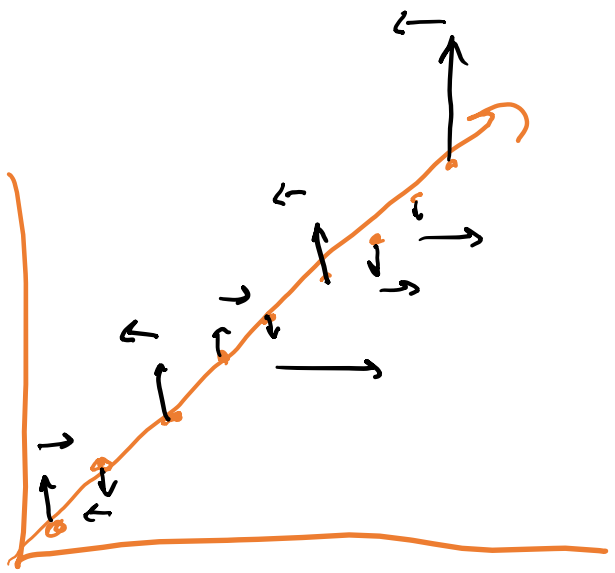


Picture Credit: Jeremy Jordan

From Output to Input Errors

- What about errors on the **input** values?

$i = 1, \dots, n: X_i \overset{iid}{\sim} P(x)$



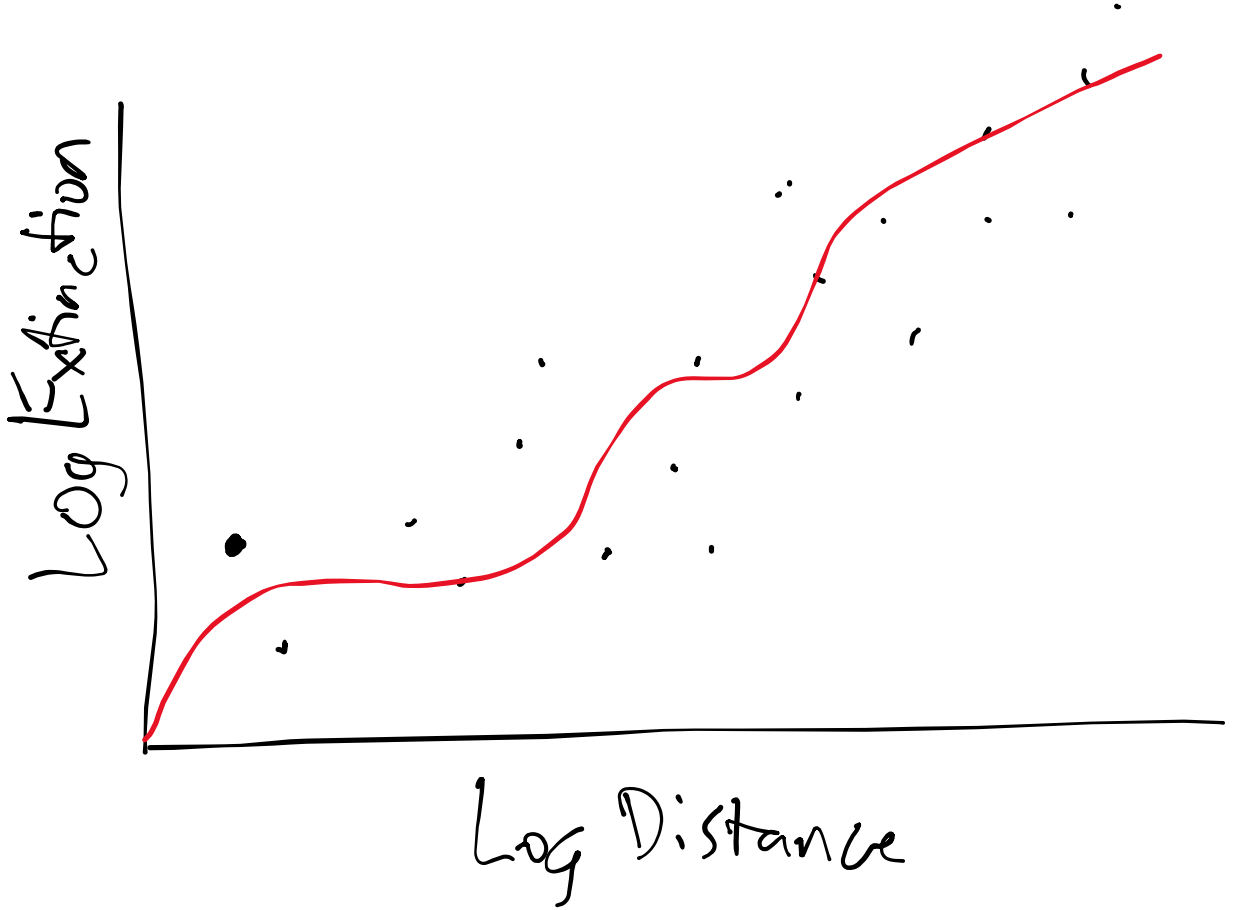
$y_i = a x_i + b$

$\tilde{x}_i = x_i + \delta_i, \quad \delta_i \overset{iid}{\sim} N(0, \sigma_{x,i})$

$\tilde{y} = y_i + \epsilon_i, \quad \epsilon_i \overset{iid}{\sim} N(0, \sigma_{y,i})$

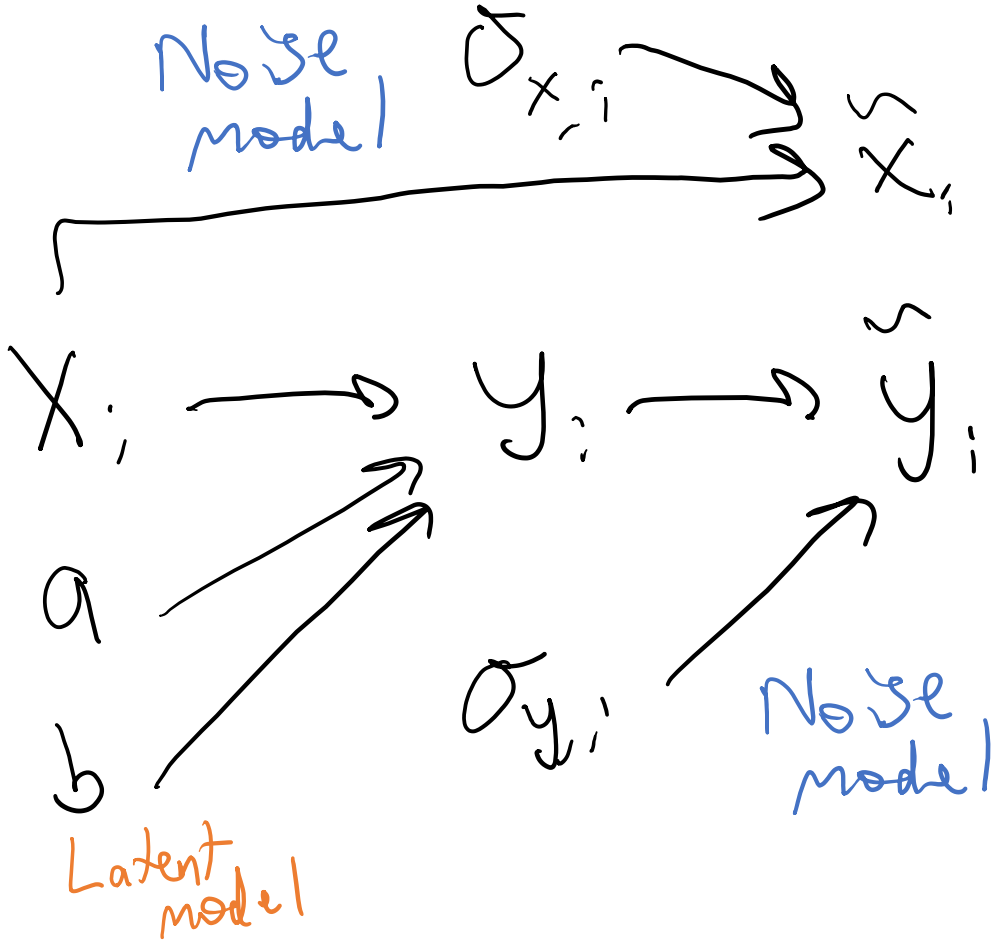
Asymmetry of Input vs Output Errors

Motivating example: **Dust mapping**



Dealing with Latent Parameters: Round 2

- Let's write down everything again…



$$P(\tilde{x}_i, \tilde{y}_i \mid \sigma_{x,i}, \sigma_{y,i}, a, b)$$

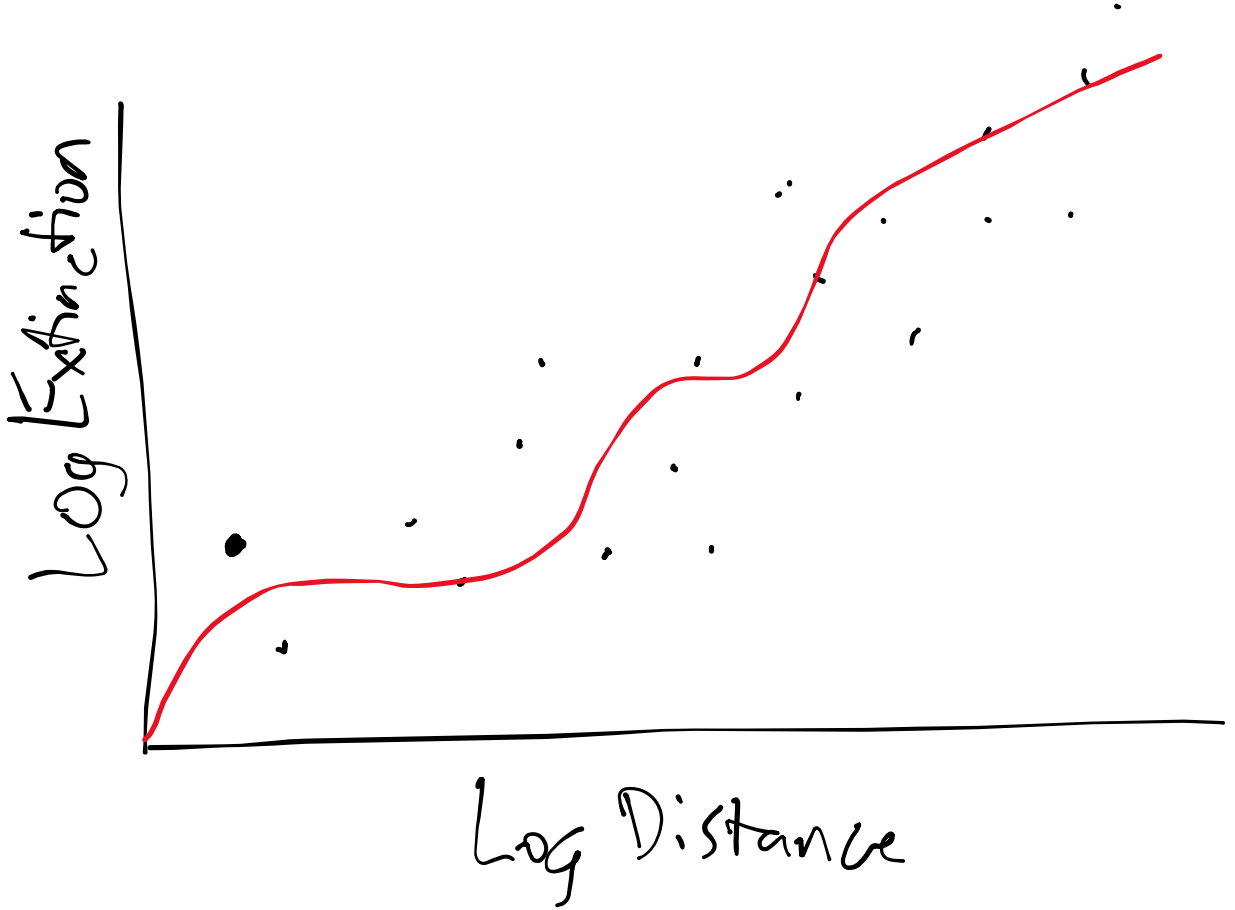
$$= \int P(\tilde{y}_i \mid y_i, \sigma_{y,i}) \times$$

$$P(\tilde{x}_i \mid x_i, \sigma_{x,i}) \times$$

$$P(y_i \mid x_i, a, b)\, dx_i\, dy_i$$

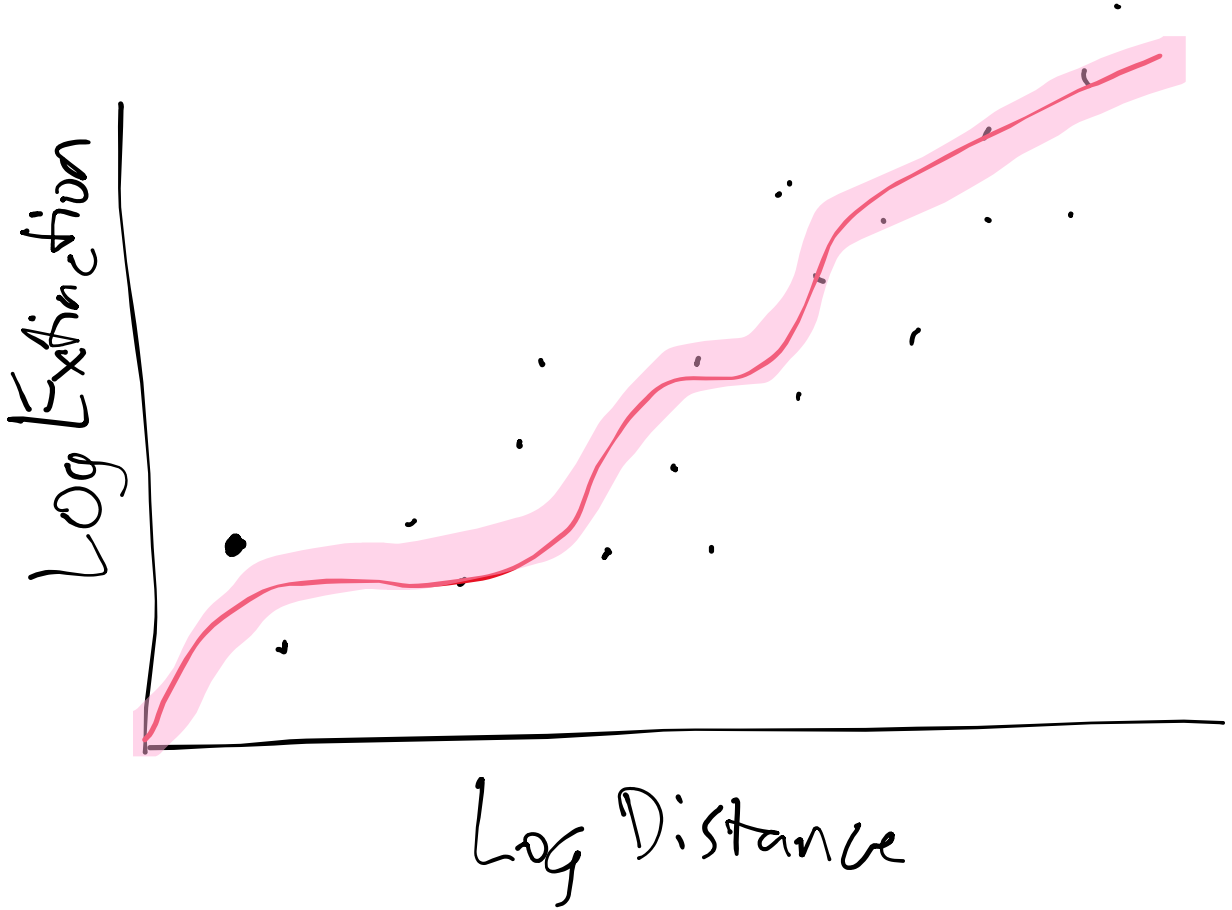
Method 1: Brute-Force Integration

Motivating example: **Dust mapping**



Method 2: Monte Carlo Integration

Motivating example: **Dust mapping**



Method 2: Monte Carlo Integration

- The math looks like:

$$P(\tilde{x}_i, \tilde{y}_i \mid \sigma_{x,i}, \sigma_{y,i}, a, b)$$

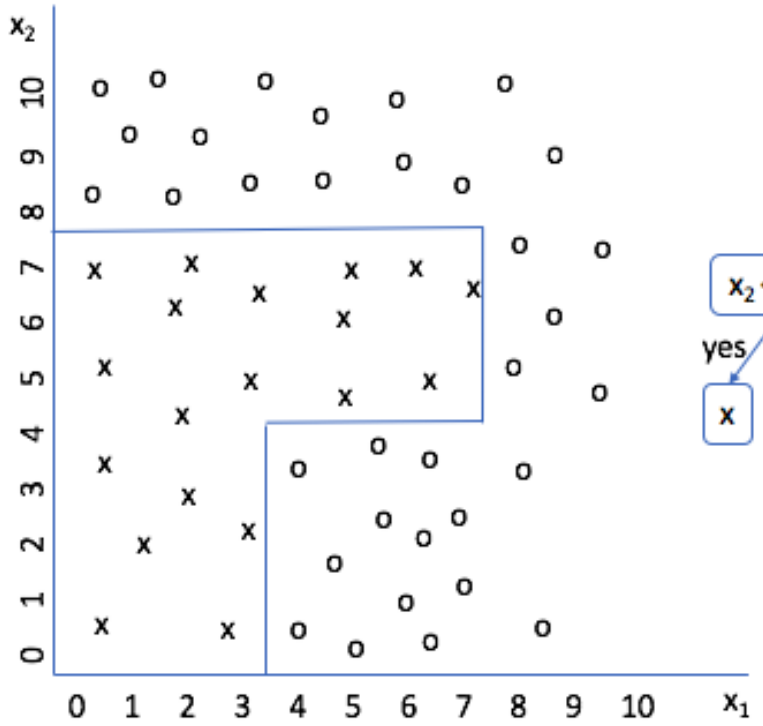
$$= \int P(\tilde{y}_i \mid y_i, \sigma_{y,i}) \times P(\tilde{x}_i \mid x_i, \sigma_{x,i}) \times P(y_i \mid x_i, a, b)\, dx_i\, dy_i$$

i.i.d samples from noise model

$$\approx \frac{1}{m} \sum_{j=1}^{m} P(y_{i,j} \mid x_{i,j}, a, b)$$

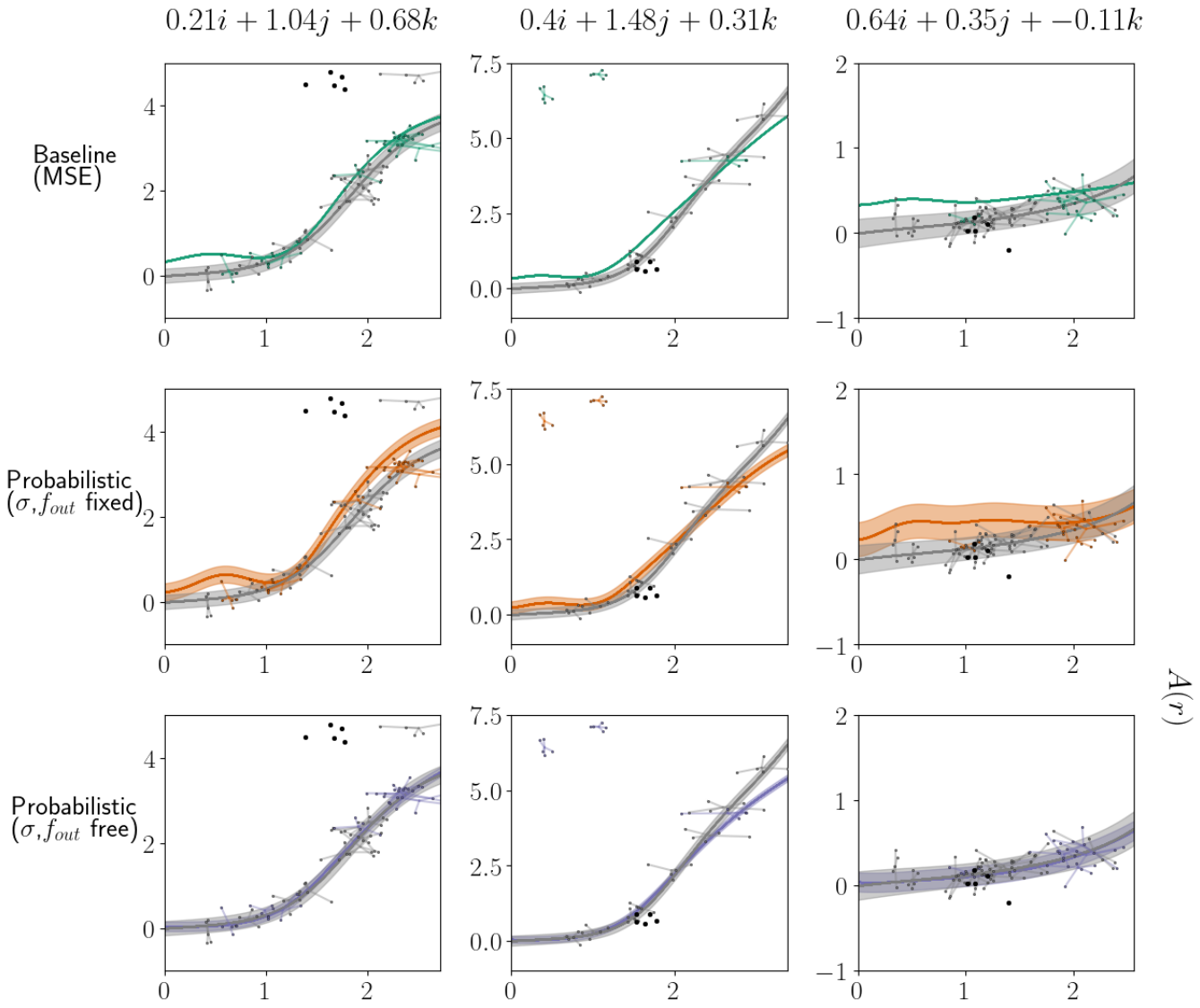
Lessons Learned: Decision Tree

- **Takeaway #3:** Monte Carlo-ing *after* training **double counts uncertainties.**
- **Takeaway #4:** Monte Carlo-ing *before* training also incorrect since **the mean of loss is not the loss of mean**.

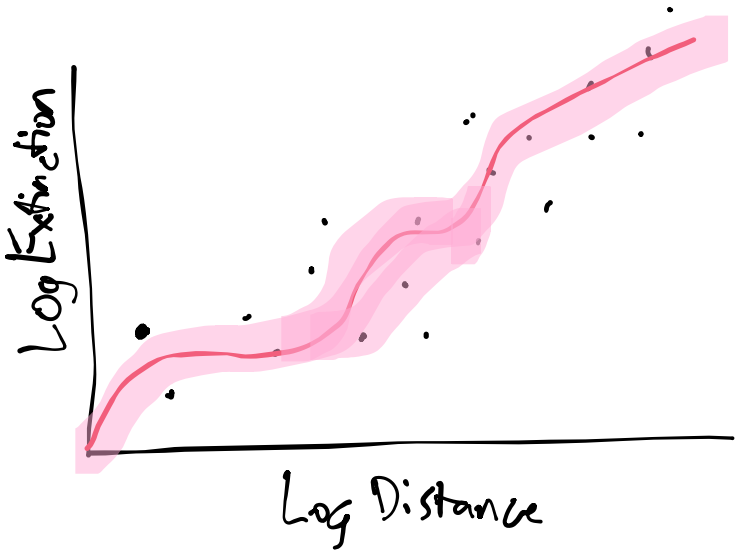


Monte Carlo training $X \rightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{pmatrix} \rightarrow \frac{1}{m}\sum L(x_j) \neq Loss(x) = -2\ln\left(\frac{1}{m}\sum P(x_j)\right)$

Implementation Example



Motivating example:
Dust mapping

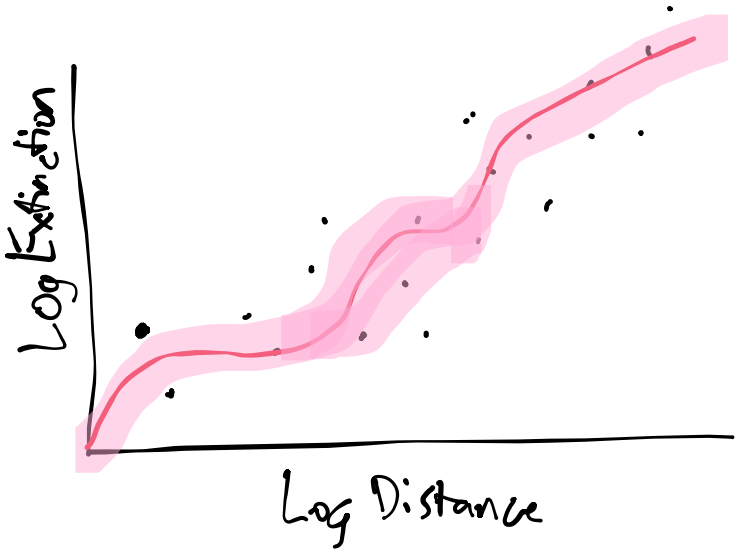


Oveisi & Speagle (in prep.)

Probabilistic Predictions?

- **Takeaway #5:** Just use **normalizing flows!**

Motivating example:
Dust mapping



Uncertainties in the Model Itself

- How do we know our optimized model with hyperparameters ϕ is the “correct” one? Should we be marginalizing over it as well? If so, how?

- **Bayesian view:** need to sample all possible model hyperparameters

- **Frequentist view:** need to sample all possible datasets

- **Takeaway #6:** Bayesian/Frequentist approaches are **not the same**.

Model Uncertainties: Bayesian

- We have a straightforward Bayesian inference problem.

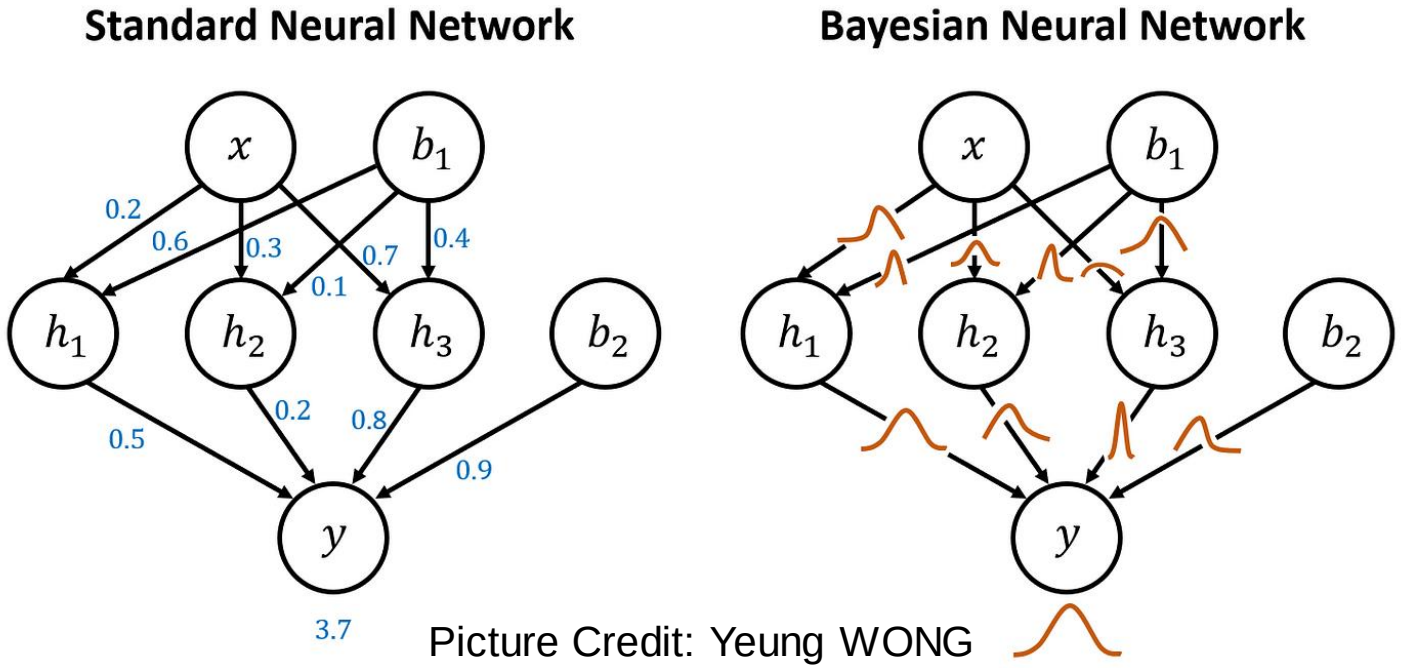
$$P(\tilde{x}_i, \tilde{y}_i | \sigma_{x,i}, \sigma_{y,i}) = \int P(\tilde{x}_i, \tilde{y}_i | \sigma_{x,i}, \sigma_{y,i}, \phi) P(\phi) d\phi$$

- Two questions to consider:
  - **What is the prior?** What does the prior even mean?
  - Do we care about **marginalizing** over the model vs **propagating** model uncertainty into the predictions? How do we do either?

$$P(\tilde{x}_i, \tilde{y}_i | \sigma_{x,i}, \sigma_{y,i}) \quad \text{vs} \quad P(\tilde{x}_i, \tilde{y}_i, \phi | \sigma_{x,i}, \sigma_{y,i})$$

Interpreting the Prior

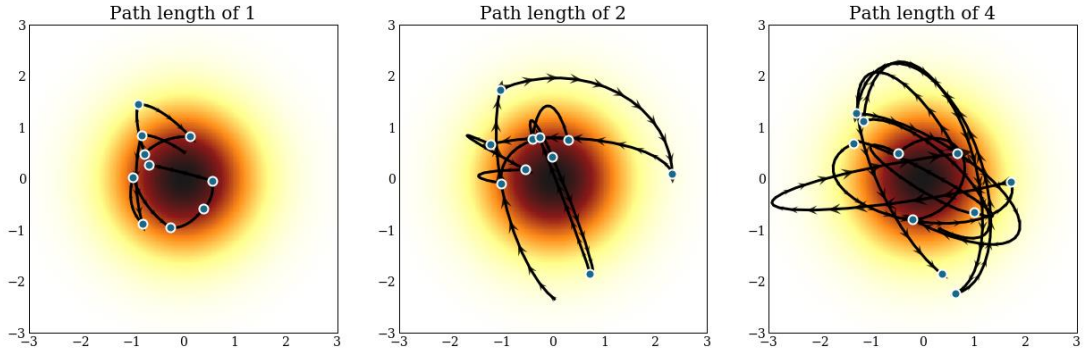
- Can be seen as a form of **regularization** – we want to set model parameters/coefficients/weights & biases to favour null values.
  - This is also used in, e.g., Bayesian polynomial regression applications where individual coefficients are set to zero.



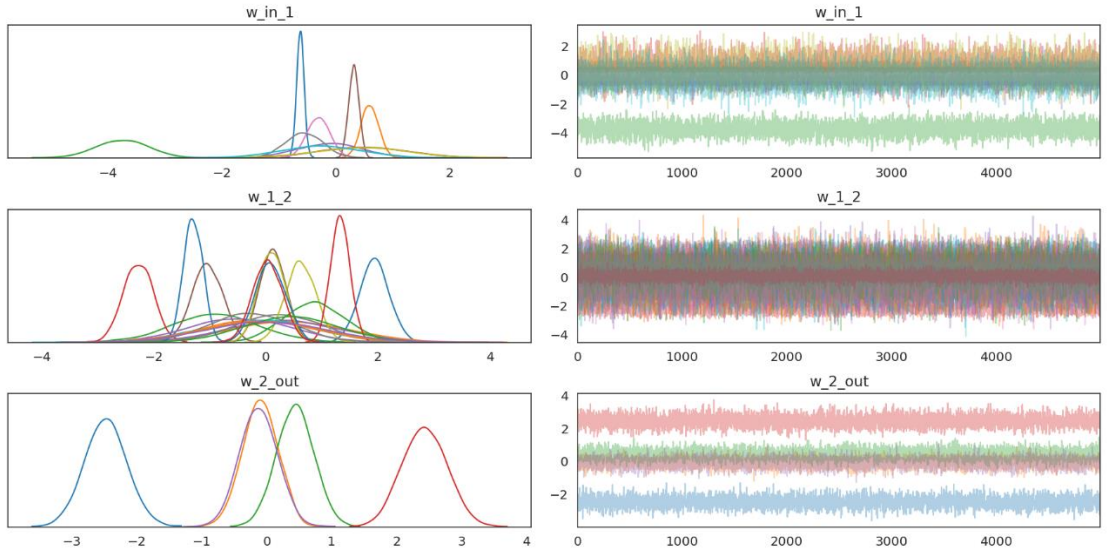
Picture Credit: Yeung WONG

Marginalizing over Parameters

- Can try to use gradient-based methods like **Hamiltonian Monte Carlo**, but it can be extremely difficult to sample parameters as the complexity of the model increases.
- Alternately, many approaches (e.g., PyMC) use **Variational Inference** to approximate the posterior distribution instead.



Picture Credit: Colin Carroll



https://www.pymc.io/projects/docs/en/v3.11.4/pymc-examples/examples/variational_inference/bayesian_neural_network_advi.html

Model Uncertainties: Frequentist

- In a Frequentist setting, we assume **there is one correct model, but many different datasets** that could arise from that model.

- **Our best guess for the model is the MLE** for ϕ given the data X, so that's what we should always use.

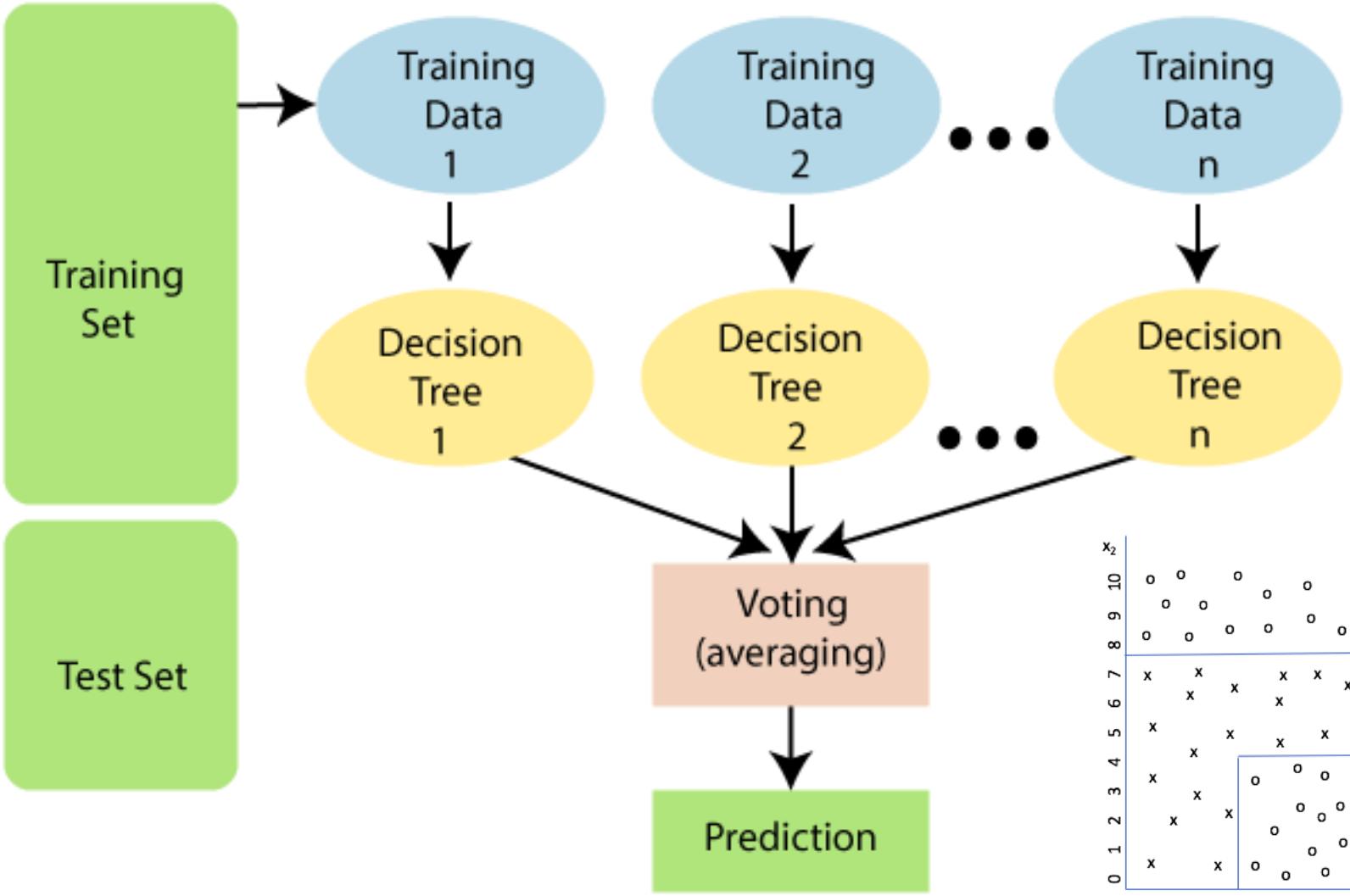
- To get uncertainties (e.g., confidence intervals), **we need to explore potential variation in the data X'.**

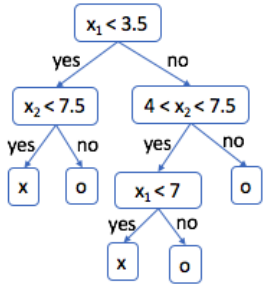
Bootstrapping and Ensembling

- We can approximate collecting new datasets using **bootstrapping (i.e. resampling with replacement)**.
- Train a new MLE model on each dataset to collect an **ensemble** of models.
- **Marginalize** by computing the **expectation value (average)** of outputs from the ensemble for a given (single) input.

- **Takeaway #7: Ensembling** leads to estimators with **Frequentist** properties (i.e. long-run odds vs certainty for a single prediction).

Example: Random Forest

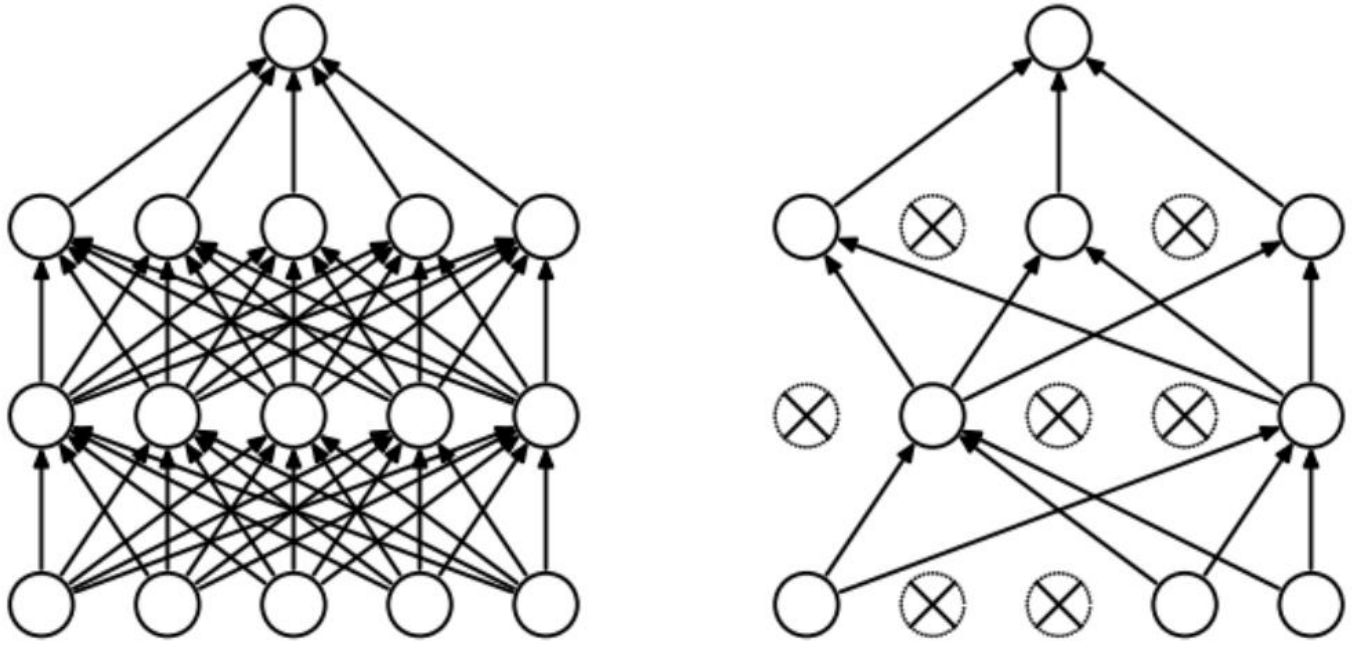




https://www.javatpoint.com/machine-learning-random-forest-algorithm

Connection to Dropout

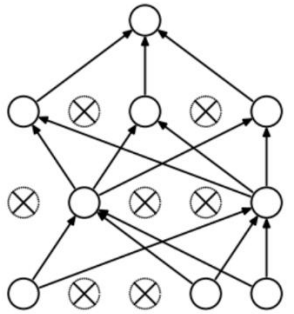
- Some neural networks exploit **dropout** to try to incorporate uncertainties. Is there a connection between dropout, ensembling (Frequentist), and/or Bayesian neural networks?



(a) Standard Neural Net (b) After applying dropout.

Srivastava et al. (2014)

Dropout as Implicit Ensembling



- Dropout gives **2^H possible neural networks** for H neurons.
  - The expected subset/architecture dependent on detailed implementation.
- **Models are correlated but can be considered as an ensemble** of (sub)networks being trained simultaneously.
  - Each unique model only sees the data once, since the probability of same model appearing twice quickly becomes negligible as H becomes large.
- If **stochastic gradient descent** is used, each model also sees a unique subset of the data.
- Final prediction computes **expectation value** across all neurons (i.e. **ensemble averaging**).

Takeaway #8: Dropout is more similar to **ensembling** over Bayesian NNs.

Validating Prediction Uncertainties

- More tricky than you might think!
  - This isn't so relevant for Bayesian methods (which are prior-dependent).
  - But it is for Frequentist ones!

- You will explore one approach in the tutorial this afternoon…

General Takeaways (Part 1)

- **The best model is a probabilistic one you can derive**
- **Monte Carlo**-ing only works if you define the right model!
  - **Logsumexp** is your friend
- **Simulation-based inference (SBI)** with **normalizing flows** is an option to completely bypass some issues
- **Model errors** can be marginalized over using **sampling** (Bayesian) or **ensembling/dropout** (Frequentist).

Uncertainty and Interpretability in Machine Learning Models

Joshua S. Speagle (沈佳士)

Department of Statistical Sciences
David A. Dunlap Department of Astronomy & Astrophysics
Dunlap Institute for Astronomy & Astrophysics
Data Sciences Institute

UNIVERSITY OF TORONTO

Overview: Part 2 (Interpretability)

- It's easy to train a complex model.
- But **we want to understand how it works** if we aim to apply it widely and across a wide variety of contexts, especially in pursuit of scientific discovery.

- Classic example: image classification! How do most models work?

Goodfellow et al. (2015)

x

"panda"
57.7% confidence

Overview: Part 2 (Interpretability)

- It's easy to train a complex model.
- But **we want to understand how it works** if we aim to apply it widely and across a wide variety of contexts, especially in pursuit of scientific discovery.

- Questions:
  - What does it even mean to **"interpret" or "explain"** an ML model?
  - Are the easy **strategies** to ensure models we train are interpretable?
  - Are there specific or general **tools** available?

Opening Discussion

- Take **a minute or two** to think of an example of an ML application in an area that you're interested in. Try to keep in mind the:
  a) Particular problem you'd try to solve
  b) The dataset you would use to train
  c) How well you think it would do overall
- You **do not** need to assume any particular method ("ML magic" is fine!)

- With the 1-2 people nearest to you, take turns explaining your idea(s) to each other.

THE Question

- Assume that **someone hands you an ML model** that they claim:
  a) Solves your exact problem
  b) Is trained on the dataset you expect
  c) Performs about as well as you'd expect overall
- They provide **no additional explanation**.

- How do you feel about this? Do you **trust** this model? If so, why? If not, what *would* it take for you to trust it?

Trust in ML

Datasheets for Datasets
(Gebru et al. 2018)

- **Transparency**
  - What exactly is the model I am using?
  - How do I know the model has been trained/validated/tested correctly?
  - Is the training data actually good?

- **Interpretability**
  - Does the model make sense? Do I understand the ideas behind it?
  - Can I find out how it works?

- **Robustness**
  - Do I know how accurate it is and under what conditions?
  - How often is it wrong? Can I find out when, how, and why?

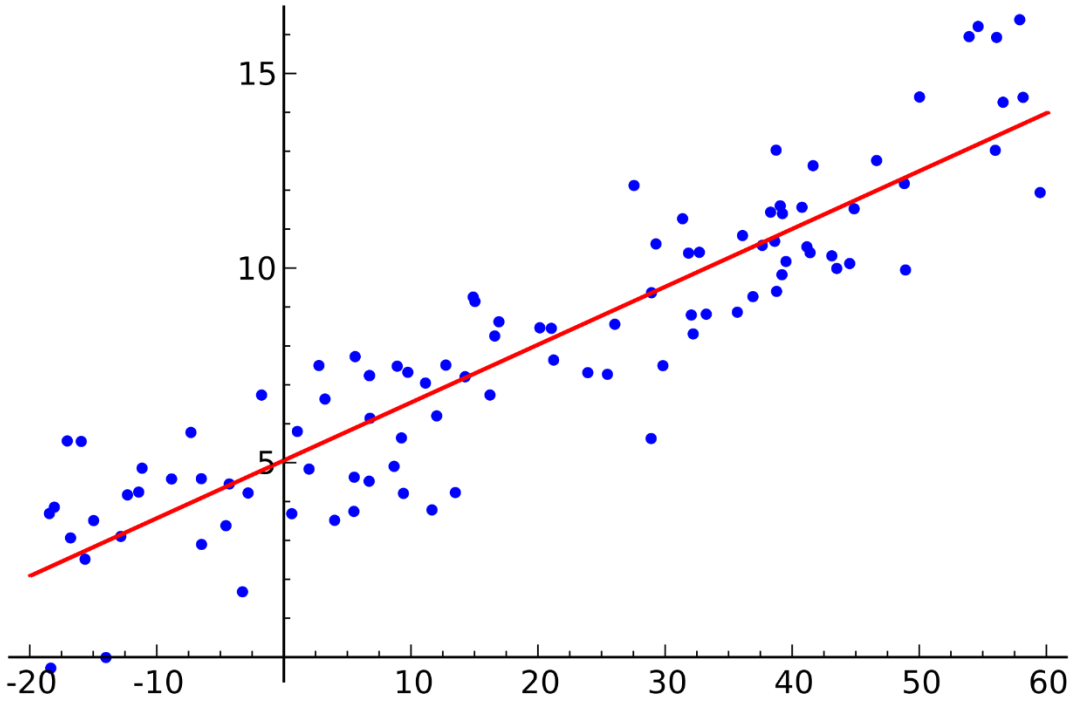
Pre-facto vs Post-facto Interpretability

- **Pre-facto:** The model is **designed** in such a way that the predictions are inherently interpretable (in whole or in part).
  - E.g., linear models, decision trees, etc.

- **Post-facto:** The model is **analyzed** using specific strategies to help **interpret** how it works **after the fact**.
  - E.g., neural networks

- **Takeaway #1:** What exactly **"interpret"** means can vary widely, regardless of whether the model is simple or complex.

Starting Point: Linear Models



Source: Wikimedia

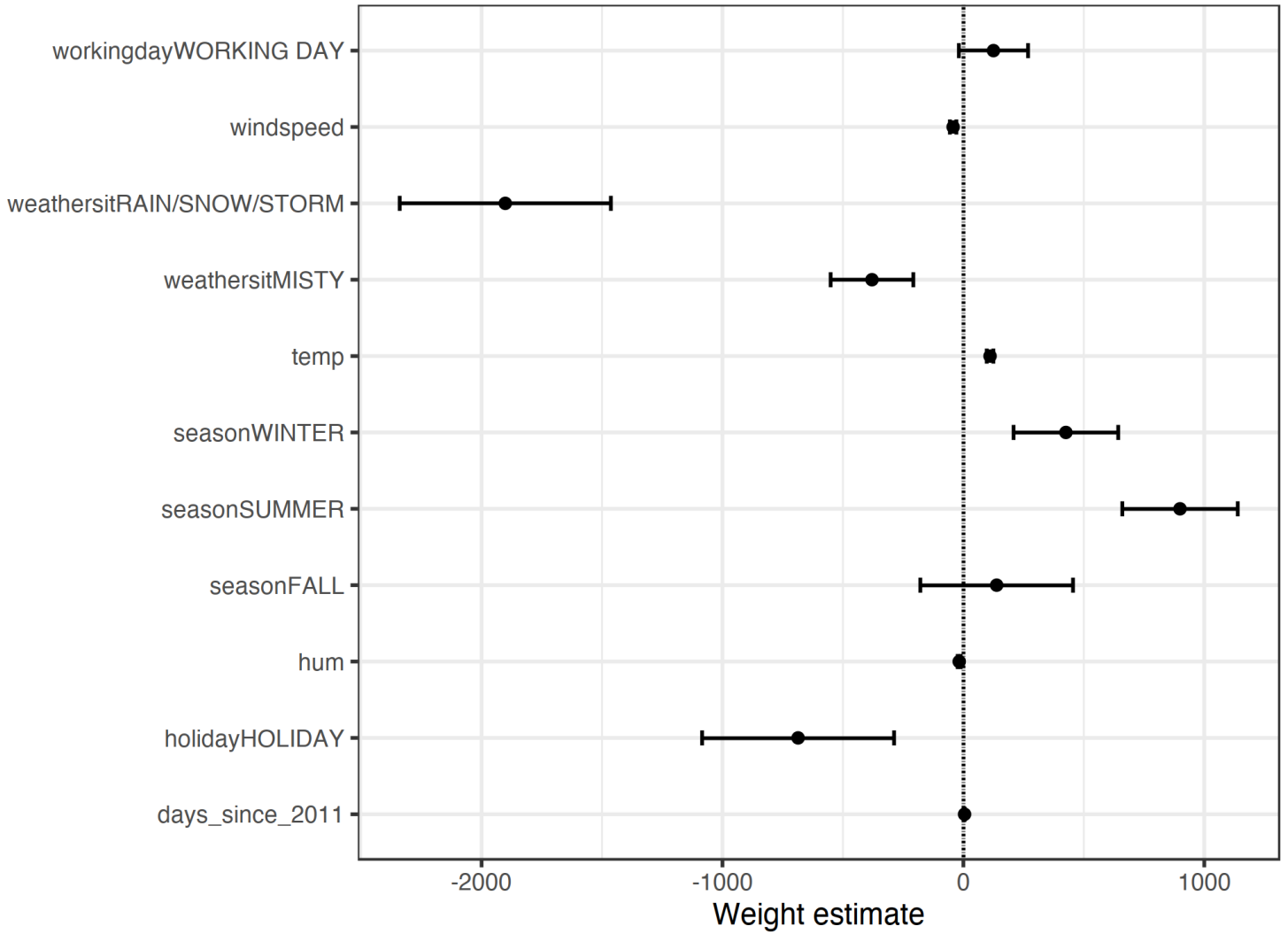
Global Feature Importance Options

1. Raw Weights

2. Error-Normalized Weights

Source: *"Interpretable Machine Learning"* by Christoph Molnar (link)

Example: Bike Rentals



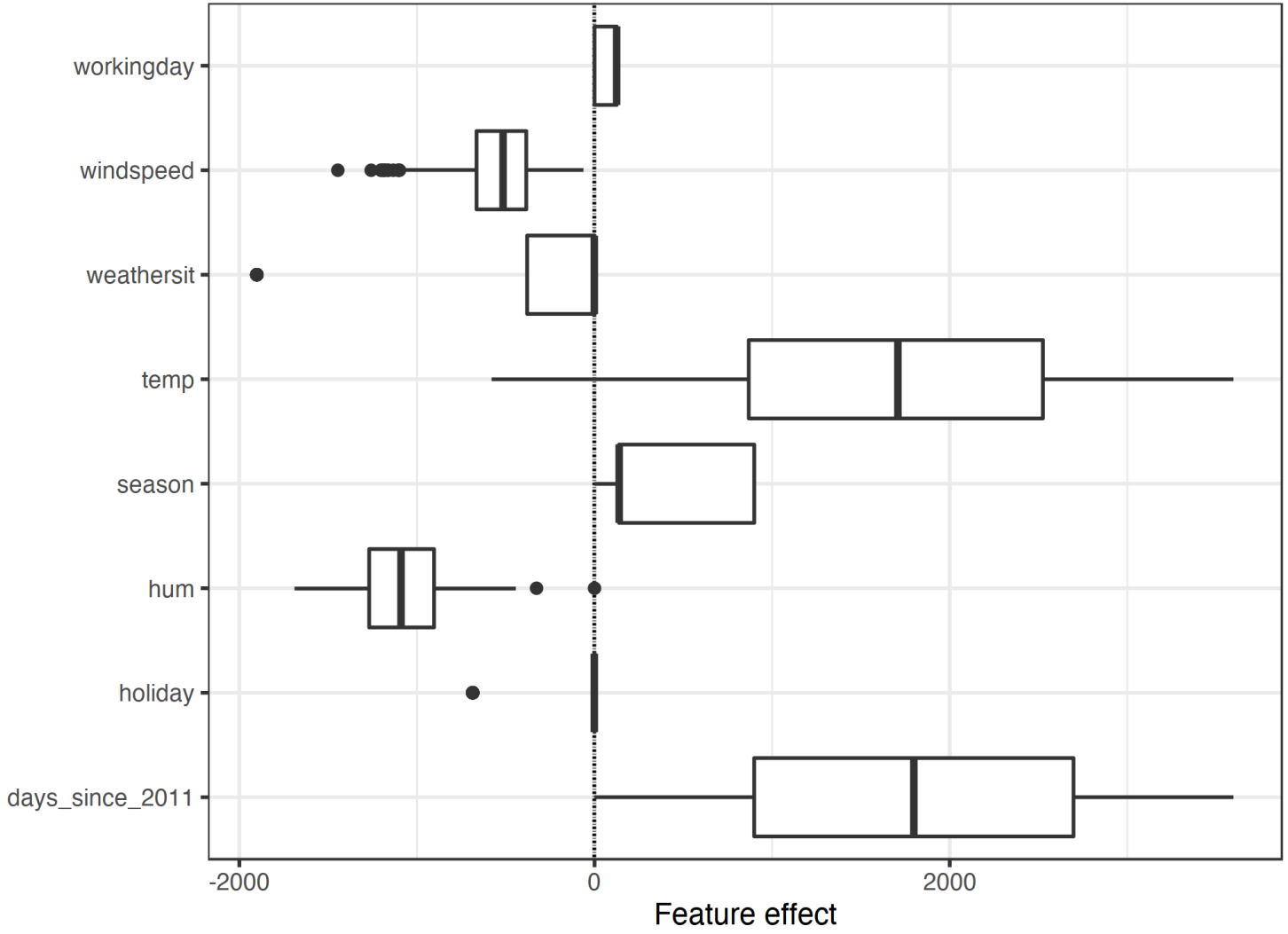
Weights

Global Feature Importance Options

1. Raw Weights
2. Error-Normalized Weights
3. Effect (Mean)
4. Effect (Standard Deviation)

Source: *"Interpretable Machine Learning"* by Christoph Molnar (link)

Example: Bike Rentals



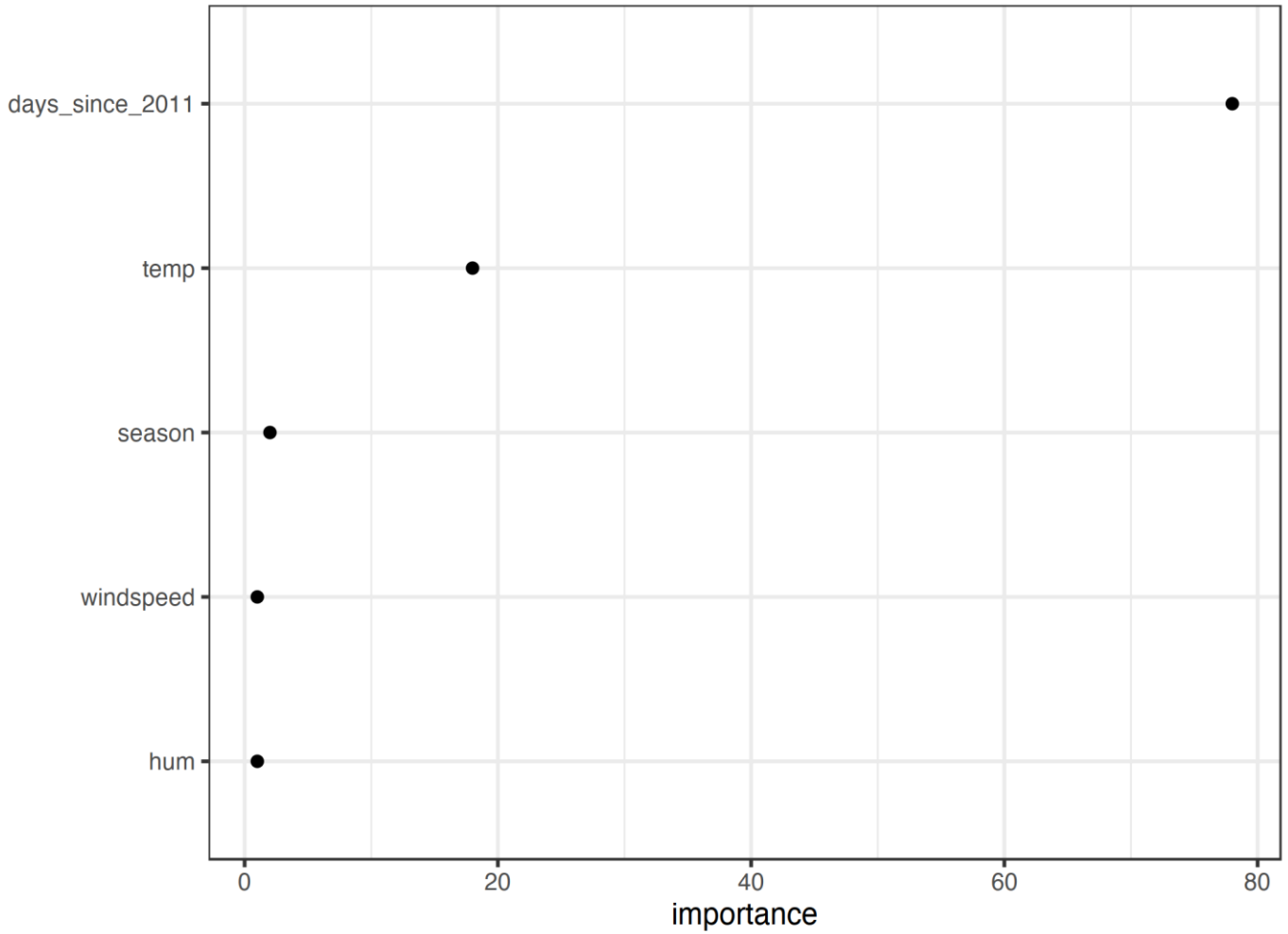
Effects

Global Feature Importance Options

1. Raw Weights
2. Error-Normalized Weights
3. Effect (Mean)
4. Effect (Standard Deviation)
5. Relative Effect Size (Mean)

Source: *"Interpretable Machine Learning"* by Christoph Molnar (link)

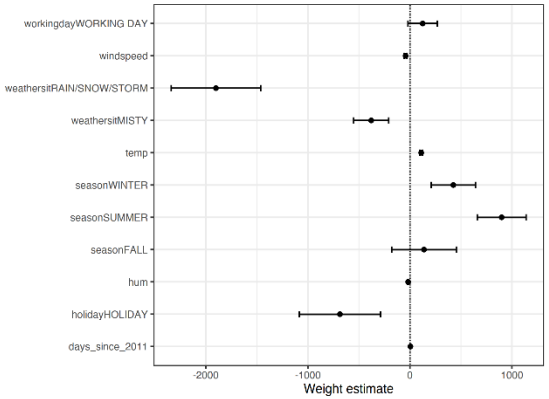
Example: Bike Rentals



Standardized
Effect Size
(Relative Importance)

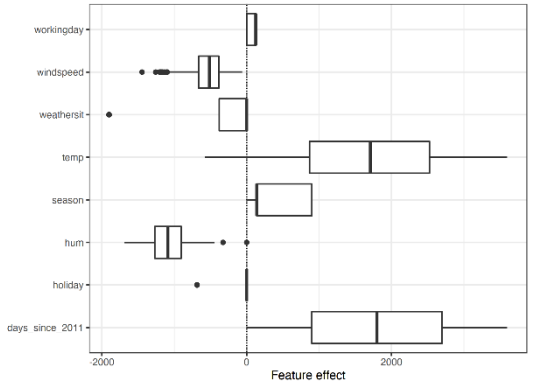
Global Feature Importance Options

1. Raw Weights



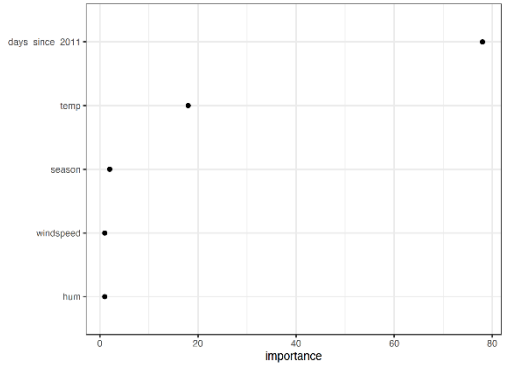
2. Error-Normalized Weights

3. Effect (Mean)



4. Effect (Standard Deviation)

5. Relative Effect Size (Mean)



What do you think? Vote on your favourite options!

Local Feature Importance Options

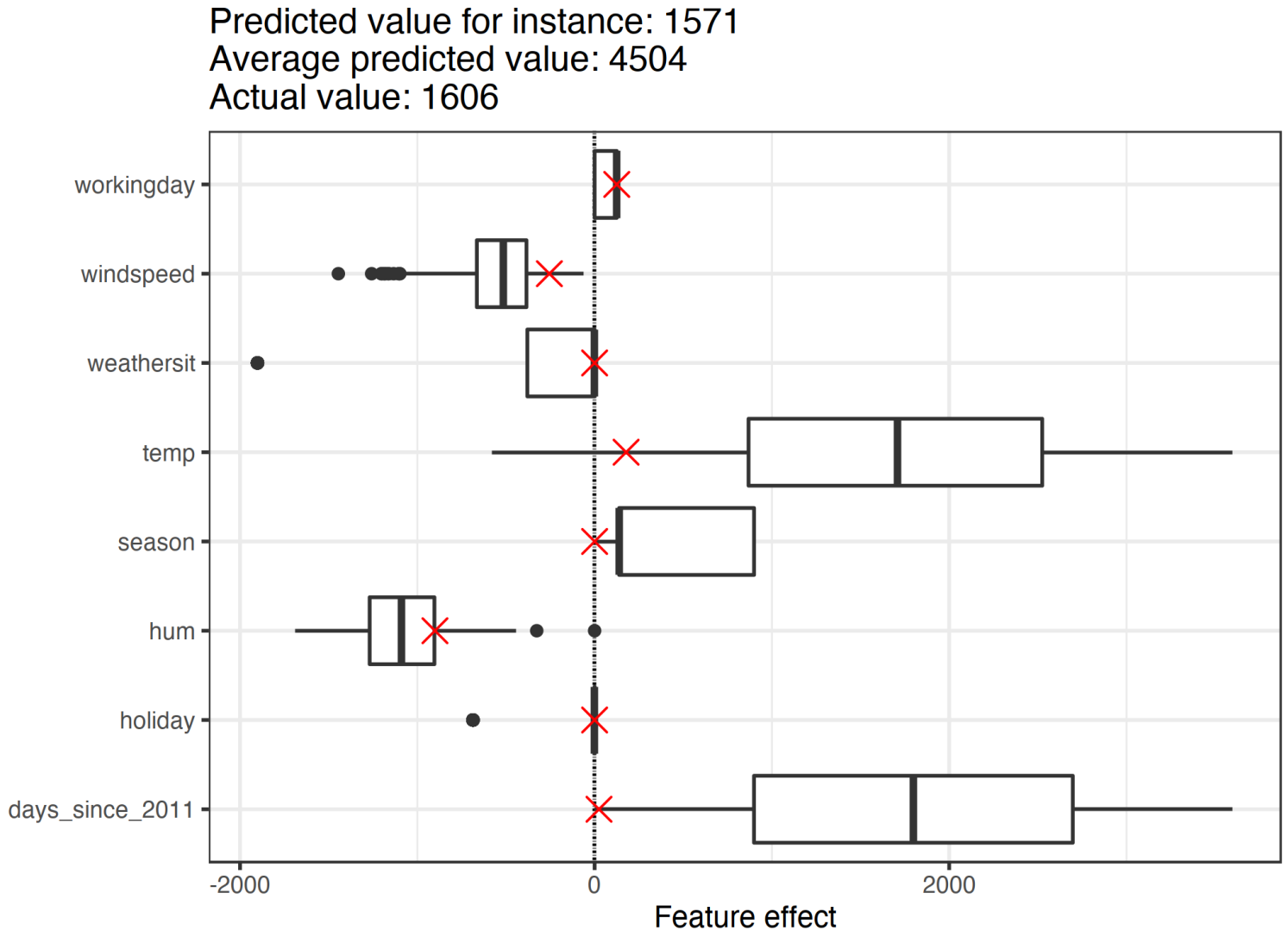
1. Raw Weights
2. Error-Normalized Weights
3. Effect (Mean)
4. Effect (Standard Deviation)
5. Relative Effect Size (Mean)

Local Feature Importance Options

1. Raw Weights
2. Error-Normalized Weights
3. **Effect** (Mean)
4. Effect (Standard Deviation)
5. **Relative Effect Size** (Mean)

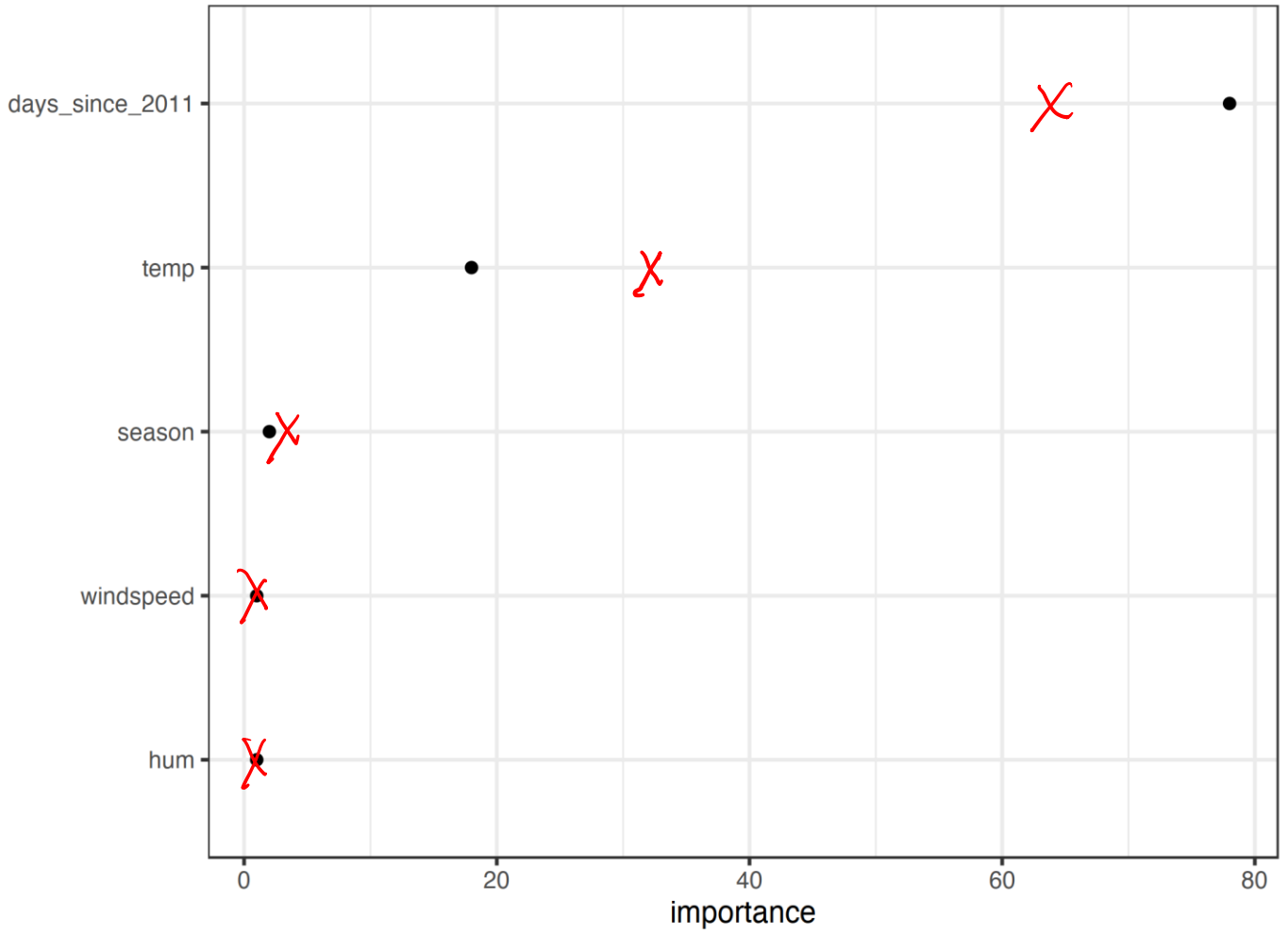
Source: *"Interpretable Machine Learning"* by Christoph Molnar (link)

Example: Bike Rentals



Source: *"Interpretable Machine Learning"* by Christoph Molnar (link)

Example: Bike Rentals



Discussion Question

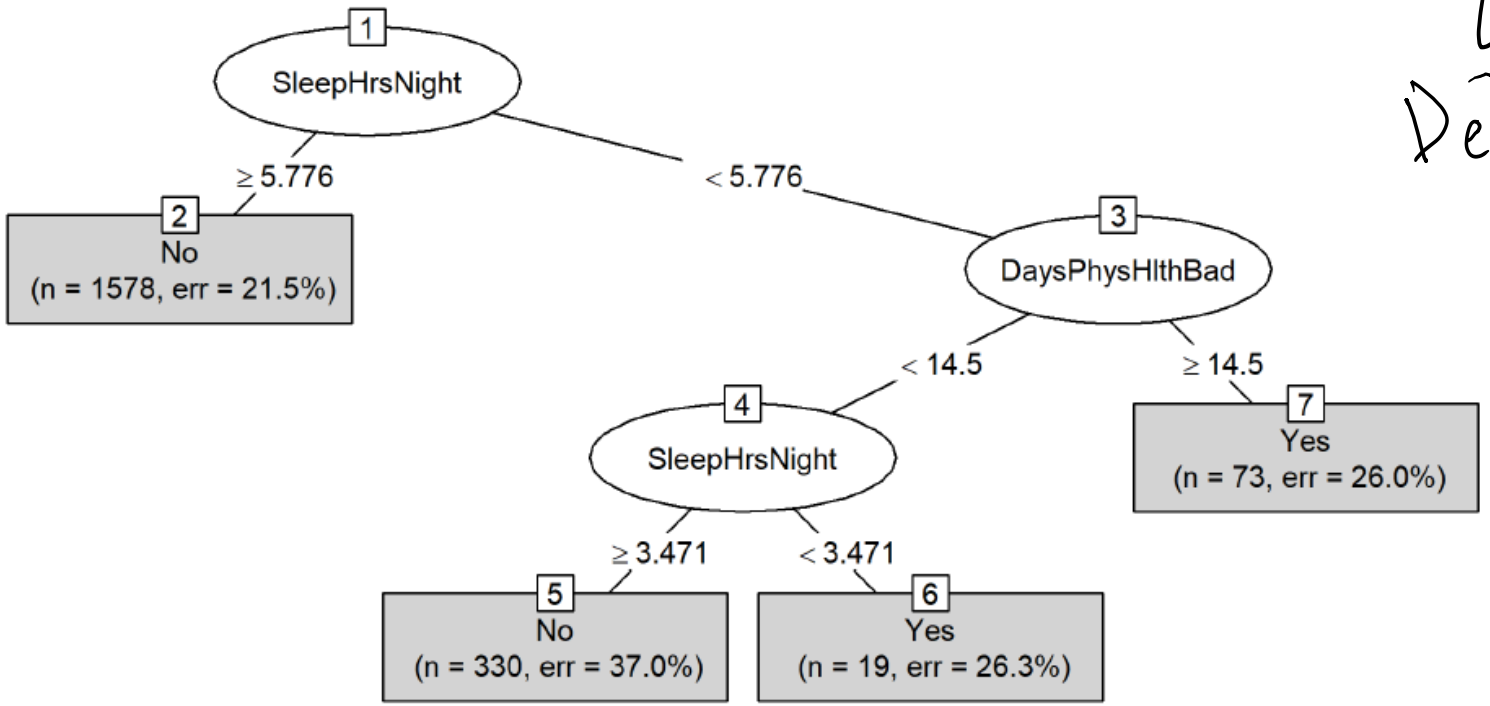
- Of the proposed methods (global vs local, coefficient vs effect vs relative effect), which one(s) do you prefer and why?

- **Takeaway #2:** Even for the most "interpretable" model, there are **many ways** to understand how and why it works.

Decision Trees: Feature Importances

- What are the **global** and **local** metrics for interpreting a decision tree model?
- What pros and cons are there for each approach?

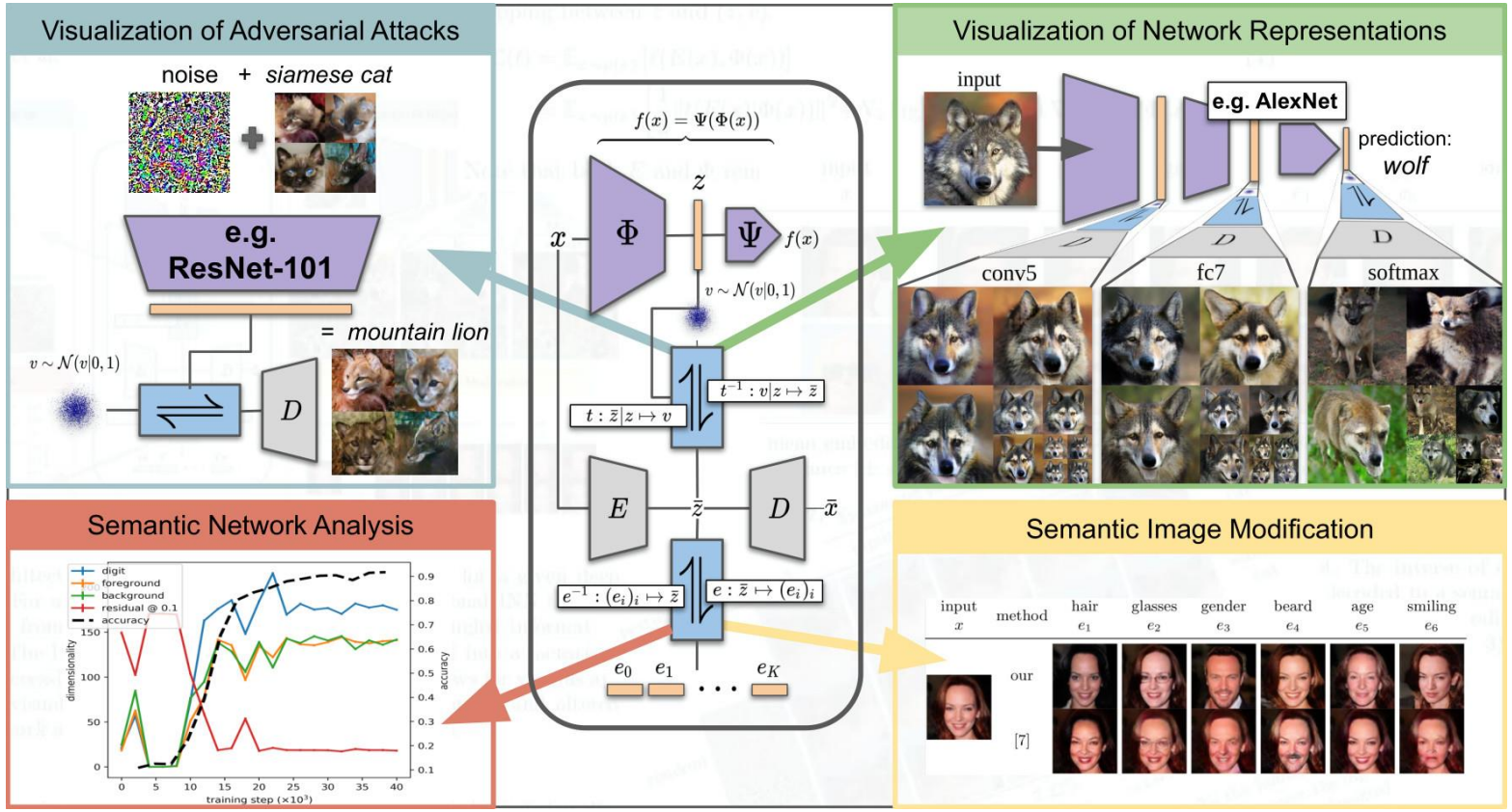
Global:
Training Splits



Local:
Decision path

Neural Network-focused Methods

- Will explore some of these in tutorials!



Rombach et al. (2020)

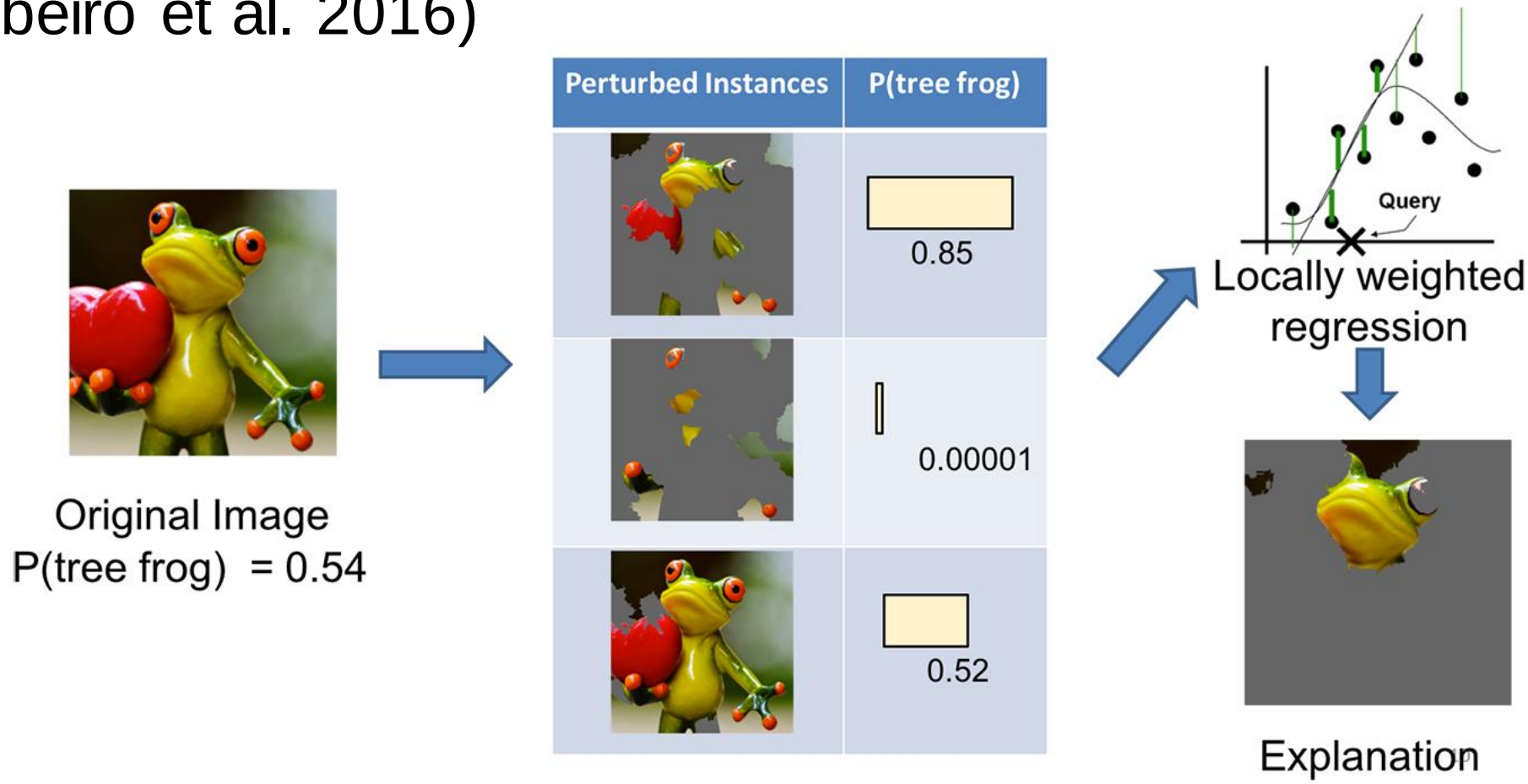
Model-Agnostic Approaches

- What **strategies** can we exploit that are not tied to a particular model?
  - (i.e. If I handed you a black box, how would you try to figure out how it worked?)

- Potential strategies:
  - Local surrogates
  - Adversarial learning
  - Model permutation/combination

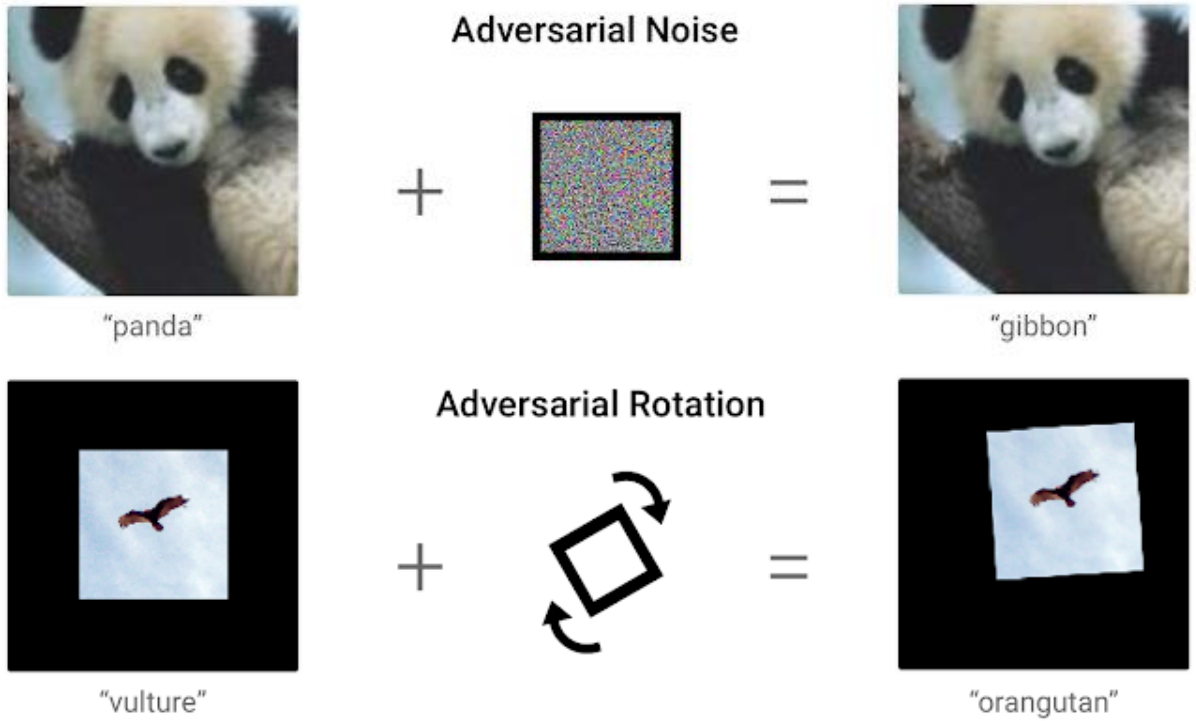
Local Surrogates

- **Local Interpretable Model-Agnostic Explanations (LIME)** (Ribeiro et al. 2016)



Adversarial/Counterfactual Examples

- Try to find the **shortest path to a decision boundary** (i.e. the normal vector) or steepest path along a regression surface.

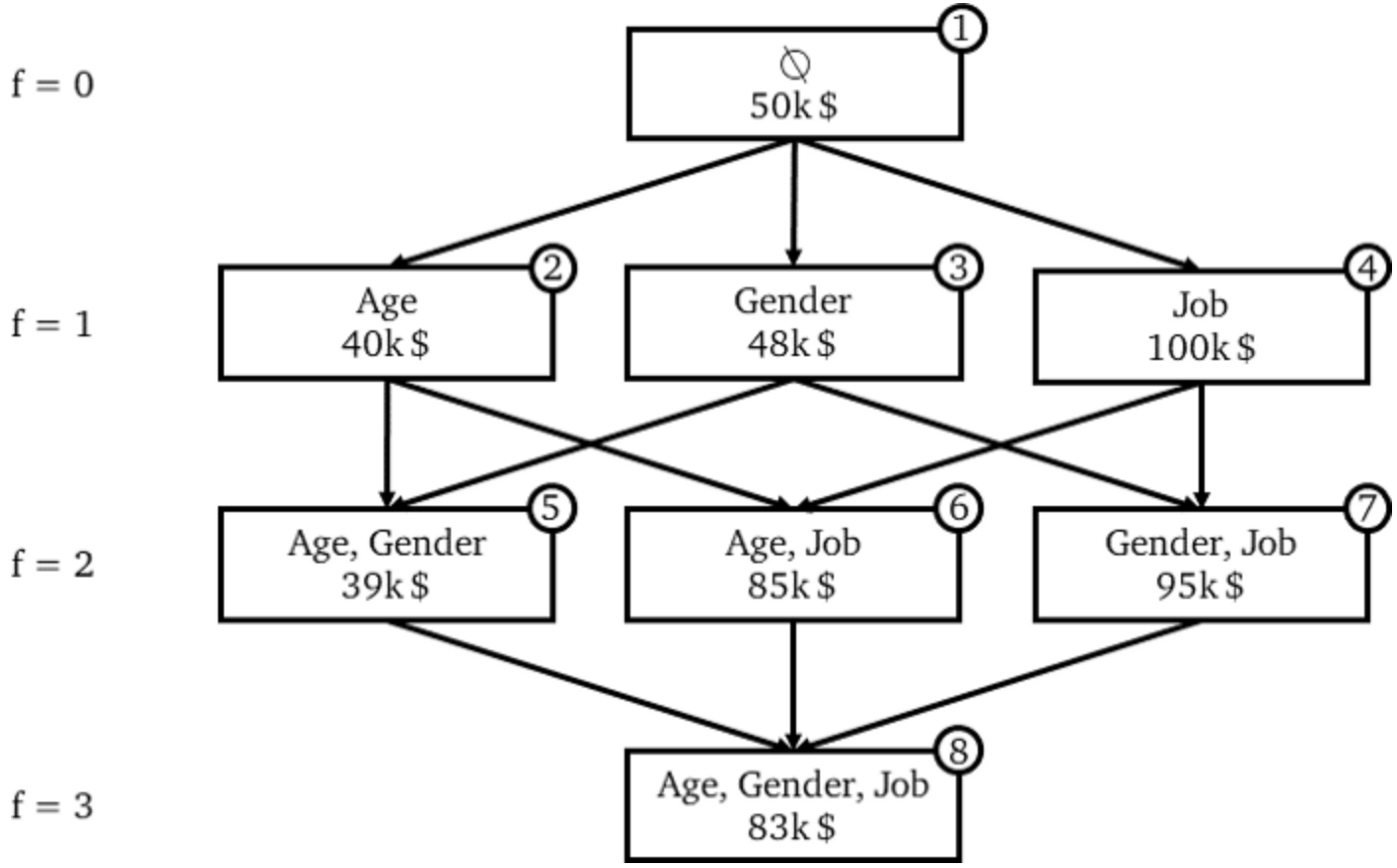


https://ai.googleblog.com/2018/09/introducing-unrestricted-adversarial.html

SHAP Values

Source: *"SHAP Values Explained Exactly How You Wished Someone Explained to You"* by Samuele Mazzanti (link)

- Shapley Additive exPlanations (SHAP; Lundberg & Lee 2017)



Discussion Question

- What **pros and cons** can you think of across the family of **model-agnostic approaches** we have discussed (surrogates, adversarial, SHAP) relative to **model-specific approaches**?

General Takeaways (Part 2)

- **There is no "best" method for interpreting a given model. Using an ensemble of different methods can help generate additional, richer insight.**
  - **Global vs local methods** tell you different information and can contextualize each others' results.
  - **Model-agnostic approaches** can be helpful for understanding complex models, provided you understand their assumptions.

- **Pre-facto interpretability is generally preferred** if you have enough time/flexibility to design a custom model.