Lecture on Machine Learning

Maria J. Molina, Assistant Professor
University of Maryland, College Park
mjmolina@umd.edu; mariajmolina.github.io

Tom Beucler, Assistant Professor
Université de Lausanne
tom.beucler@unil.ch; wp.unil.ch/dawn/

UMD PARETO group

∂³AWN
∂ata-∂riven Atmospheric & Water ∂yNamics

Learning Outcomes (Profs. Molina and Beucler)

1. Frame an ML problem (e.g., regression or classification) for your specific application.
2. Choose appropriate data preprocessing steps.
3. Understand how ML models are trained.
4. Define physics-guided ML and apprehend the diversity of available approaches.

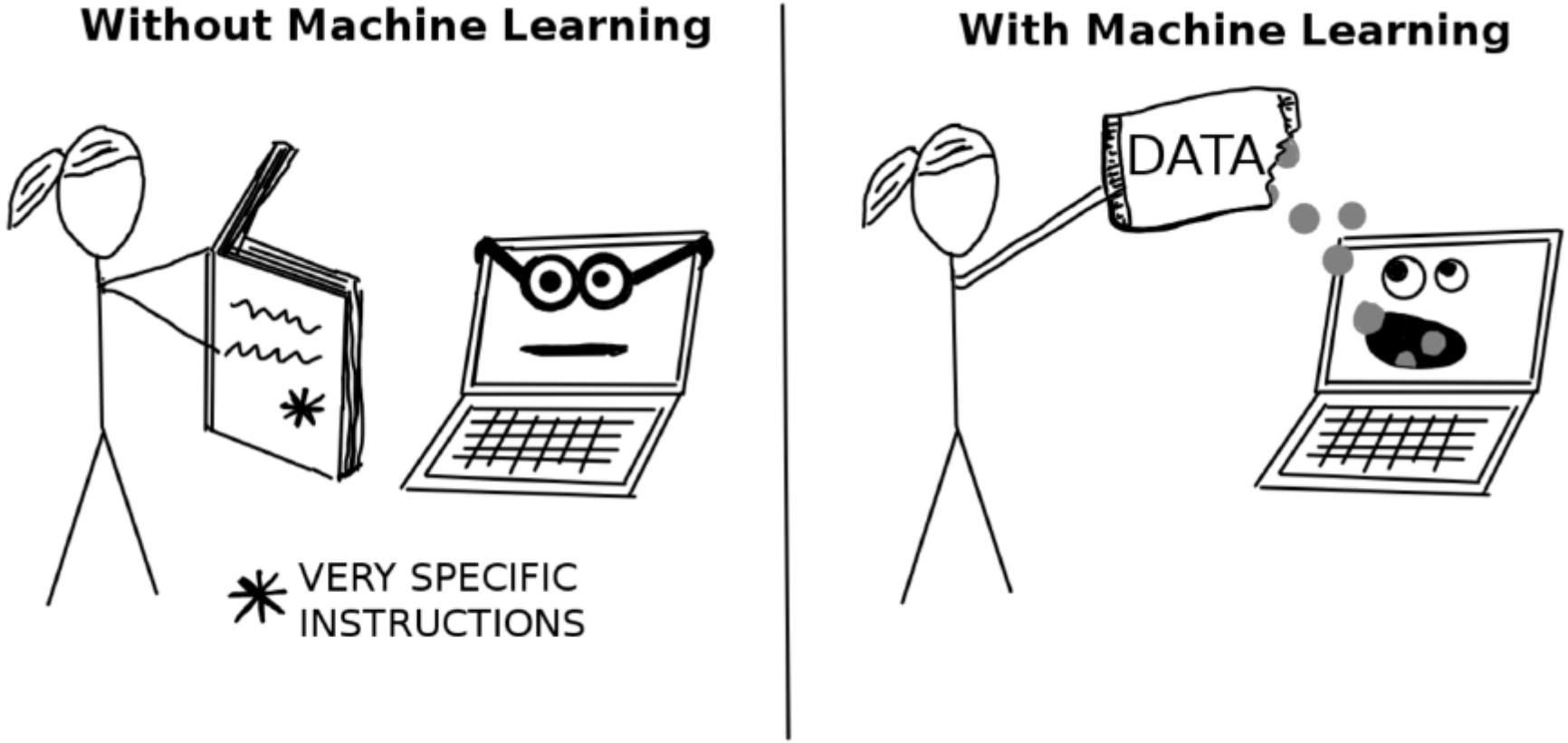
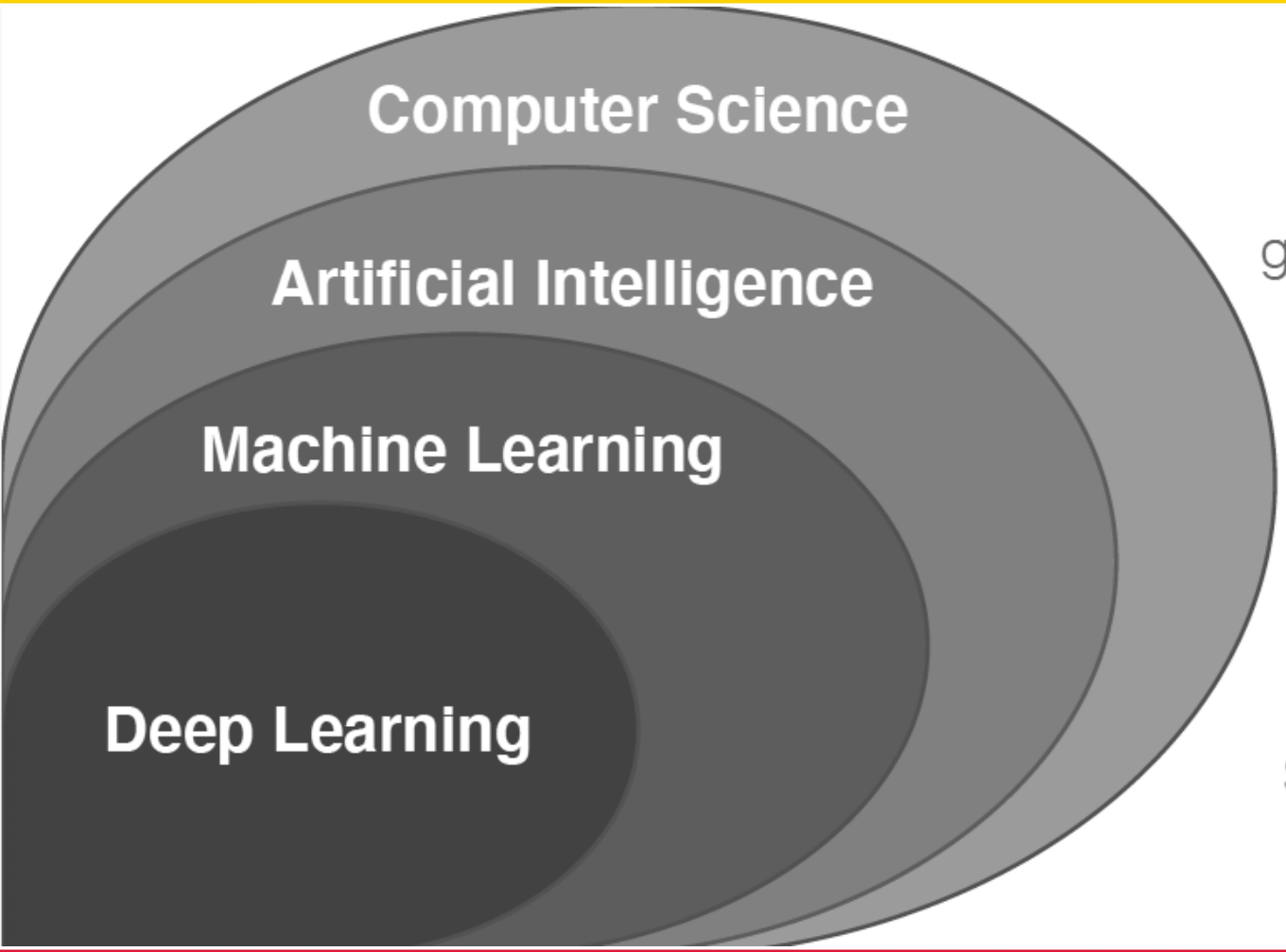
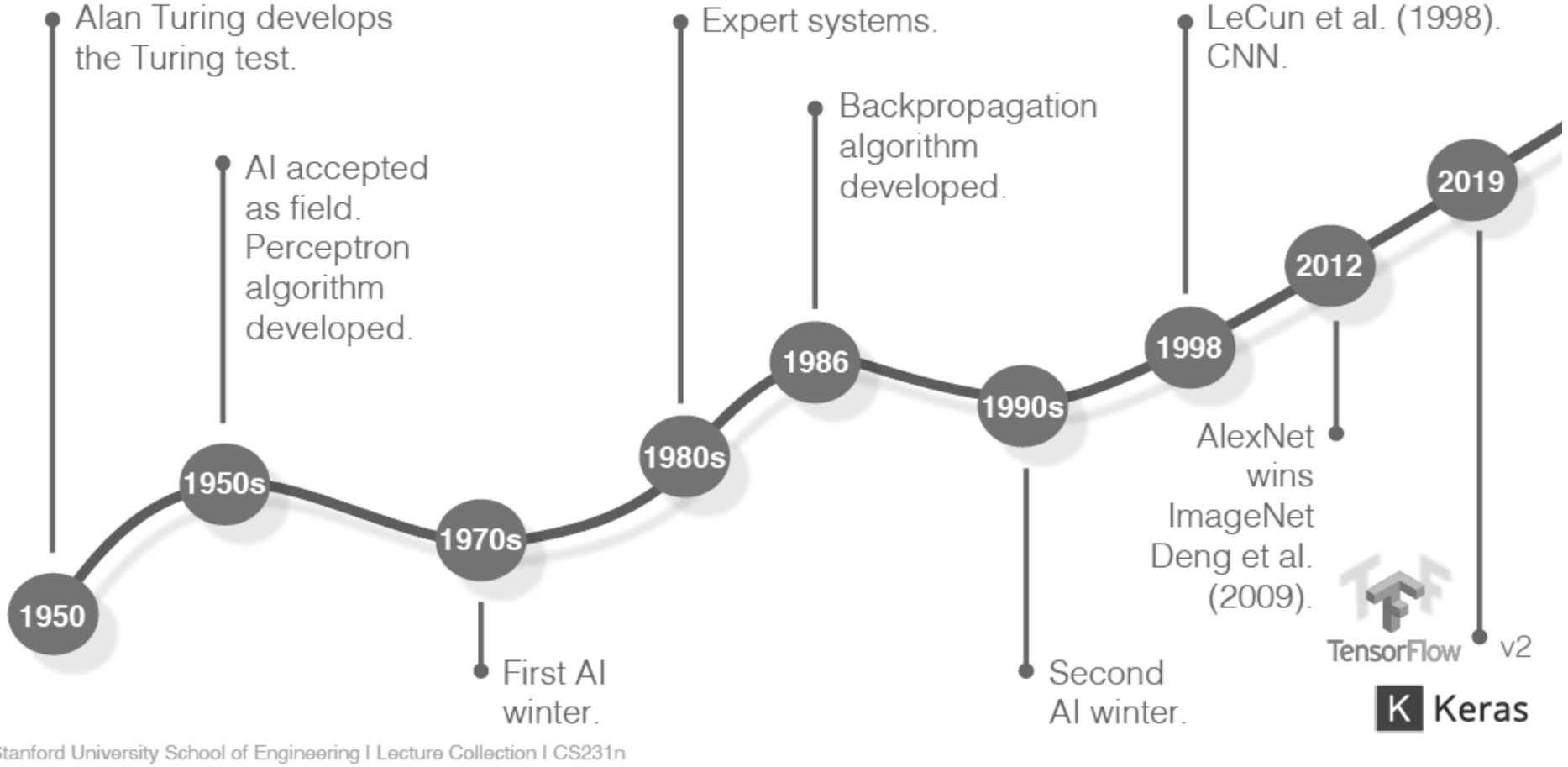


Figure from Interpretable Machine Learning (Molnar, 2019).



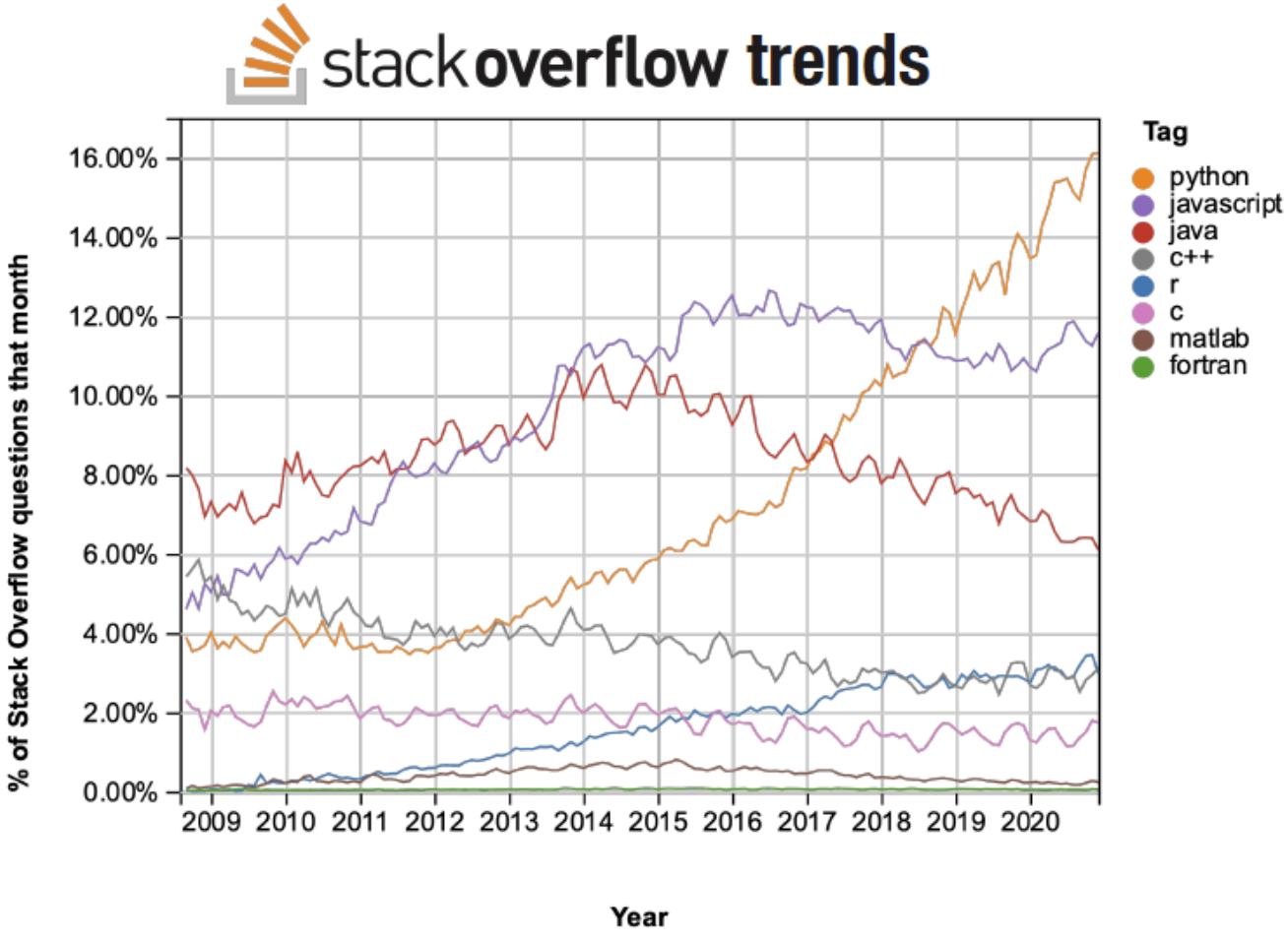
"Field of study that gives computers the ability to learn without being explicitly programmed."

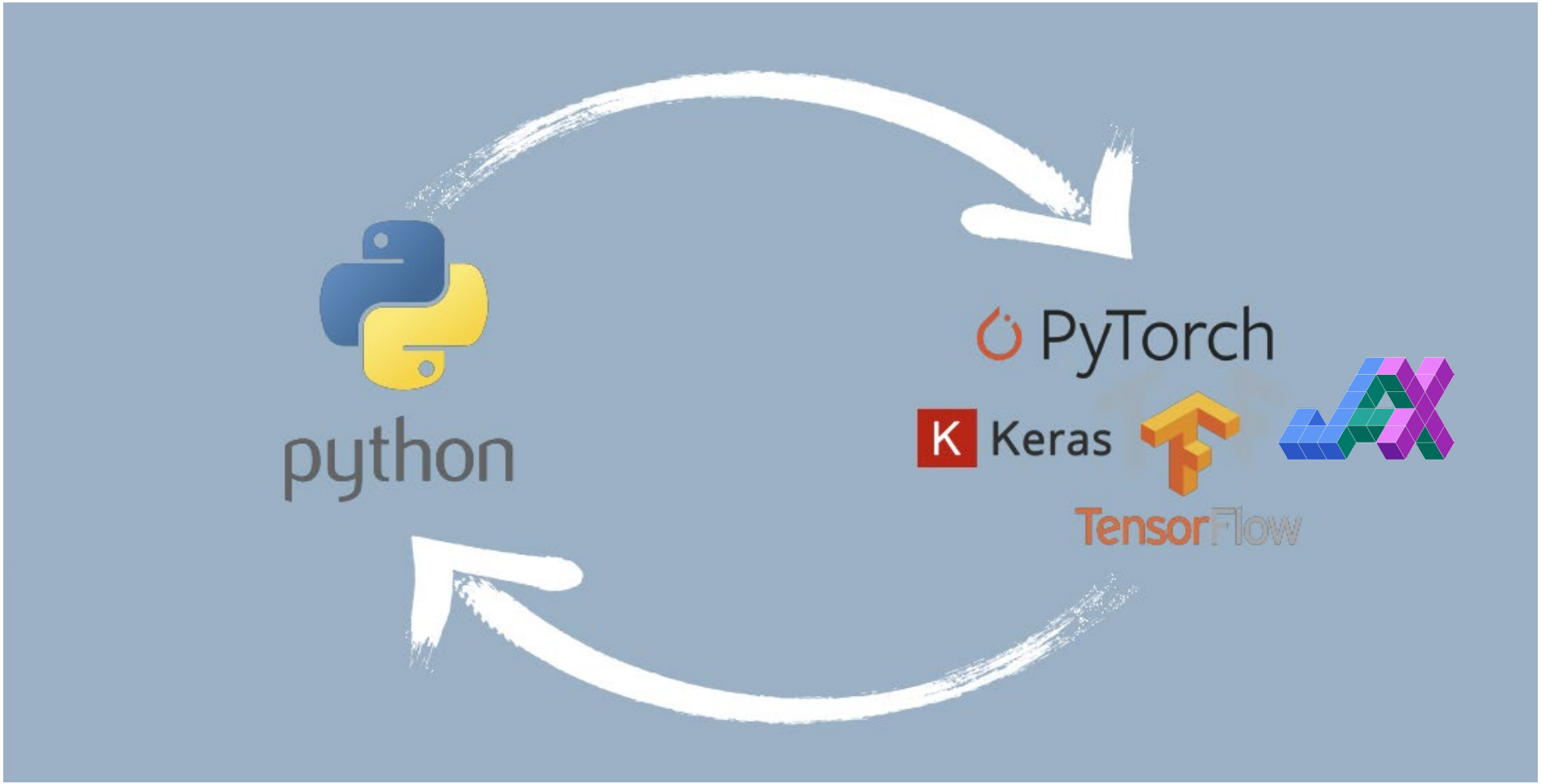
1959, Arthur Lee Samuel defined ML

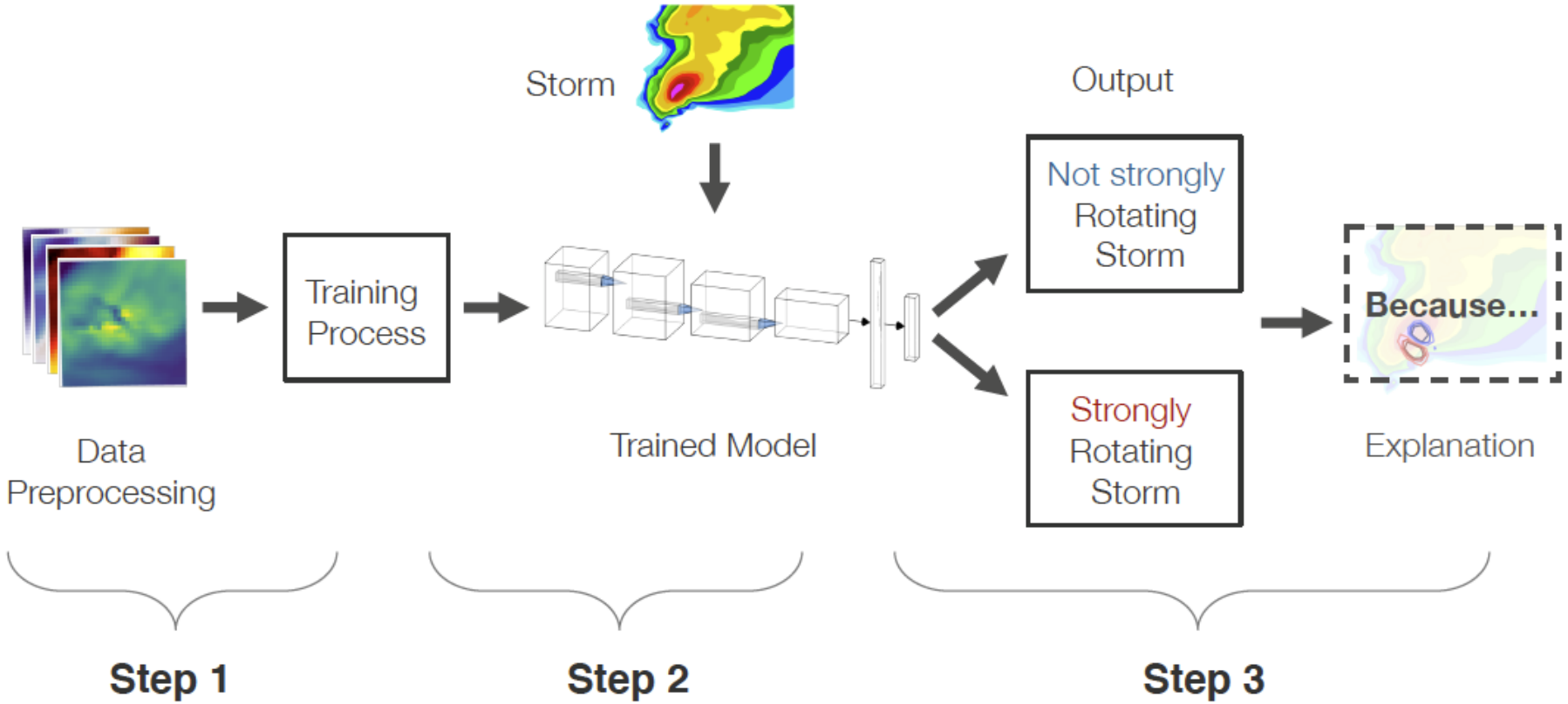

Alan Turing develops the Turing test.
AI accepted as field. Perceptron algorithm developed.
Expert systems.
Backpropagation algorithm developed.
LeCun et al. (1998). CNN.
1950
1950s
1970s
1980s
1986
1990s
1998
2012
2019
First AI winter.
Second AI winter.
AlexNet wins ImageNet Deng et al. (2009).
TensorFlow
v2
K Keras

How did we get here?

1. Large, high-quality and labeled datasets
2. GPUs (graphics processing units)
3. Benchmarking competitions and algorithmic advances
4. And I argue… open source software


stackoverflow trends
% of Stack Overflow questions that month
16.00%
14.00%
12.00%
10.00%
8.00%
6.00%
4.00%
2.00%
0.00%
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
Year
Tag
python
javascript
java
c++
r
c
matlab
fortran


python
PyTorch
K Keras
TensorFlow


Storm
Output
Not strongly Rotating Storm
Strongly Rotating Storm
Training Process
Because...
Data Preprocessing
Trained Model
Explanation
Step 1
Step 2
Step 3

Take a guess…

In your machine learning project, how much time will you spend preparing and transforming data?

a. Less than half the time

b. Half the time

c. More than half the time

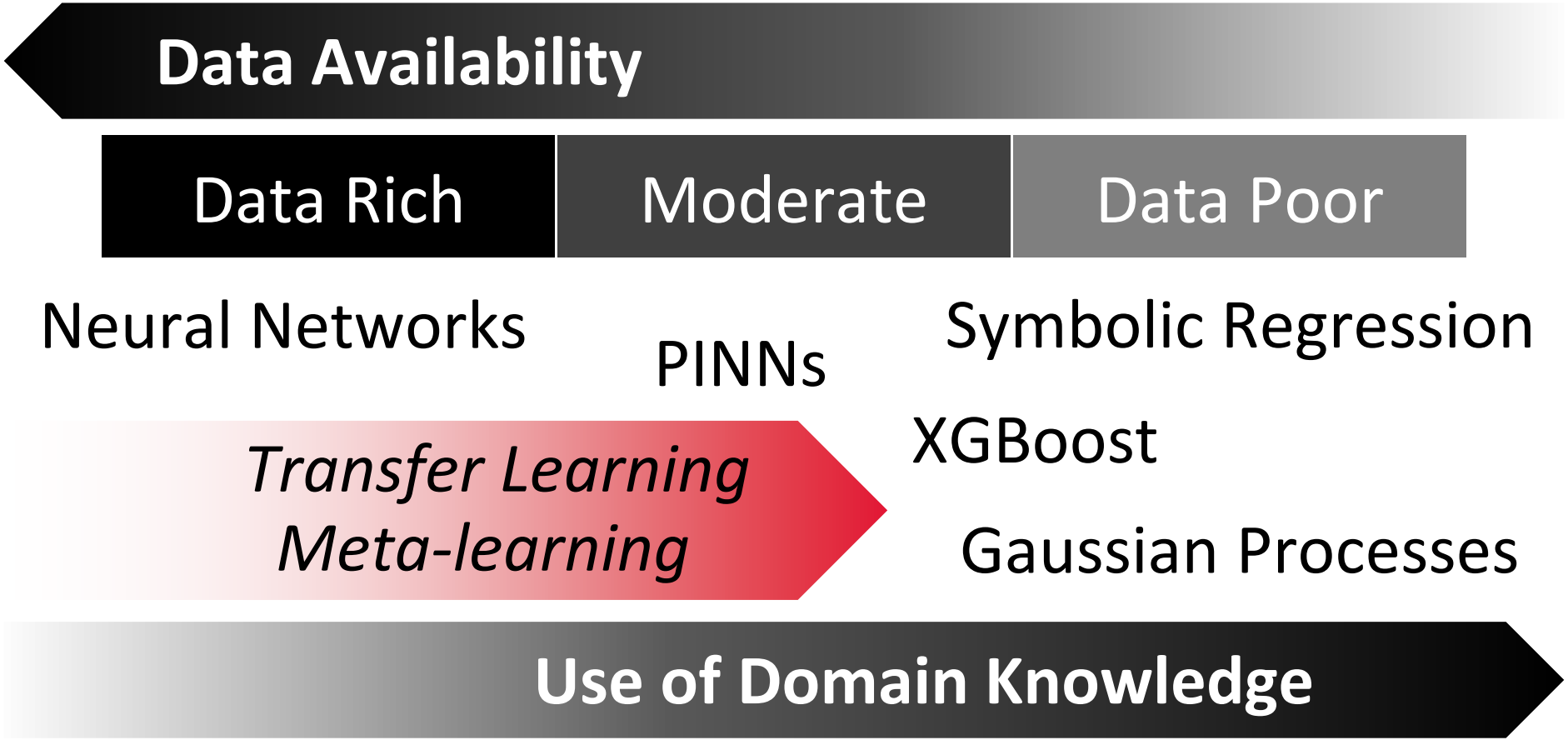
Take a guess…

In your machine learning project, how much time will you spend preparing and transforming data?

a. Less than half the time

b. Half the time

c. More than half the time



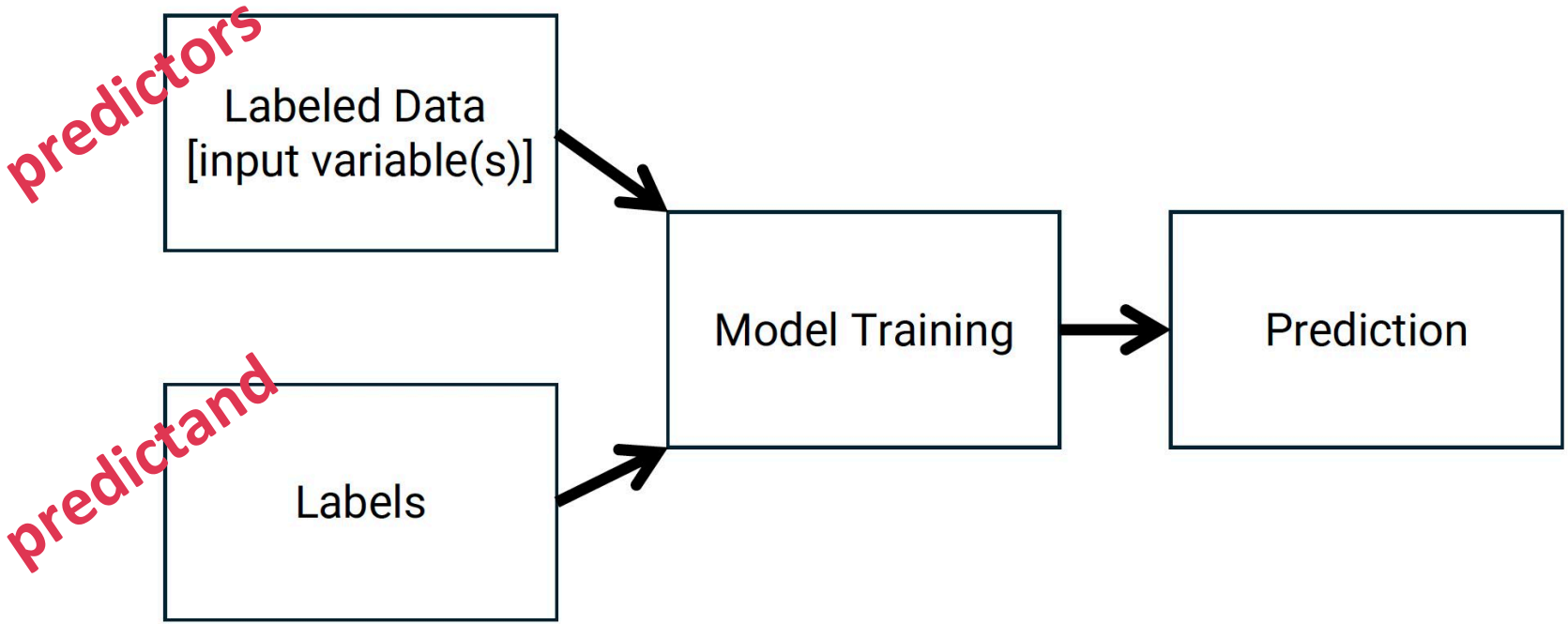
Adapted and Inspired from

Supervised Learning

Unsupervised Learning

Week 2	**Data exploration, preprocessing, labels, a**
Week 3	**Introduction to artificial neural networks**
Week 4	**Hyperparameter optimization of your neu**
Week 5	**Convolutional neural networks** (when spat
Week 6	**Recurrent neural networks** (when tempora
Week 7	**Self-organizing maps**
Week 8	**Autoencoders**

Supervised Learning: we know the answers beforehand.

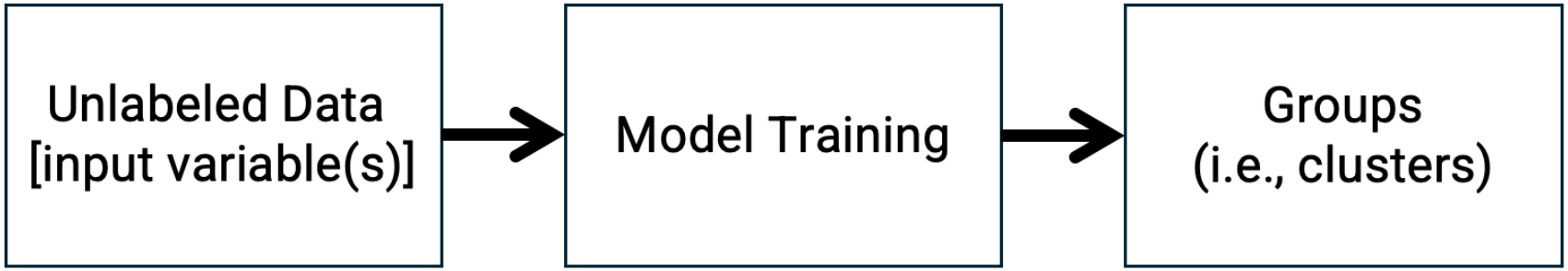


Supervised Learning: we know the answers beforehand.

$y = mx + b$

y are your labels; x is your input

Unsupervised Learning: we don’t “know” the answers beforehand.



Unsupervised Learning: we don’t “know” the answers beforehand.

$y = mx + b$

We have no labels (y). Usually used for grouping data or dimensionality reduction.

Semi-supervised Learning

can have unpaired data $\{x_1, x_2, ...\}$ and labels $\{y_1, y_2, ...\}$ and a small set of paired data $\{(x_1,y_1), (x_2, y_2) ...\}$.

Self-supervised Learning

can sometimes be just using $\{x_1, x_2, ...\}$ without any labels $\{y\}$, e.g., learn an useful representation of images, before training a classifier.

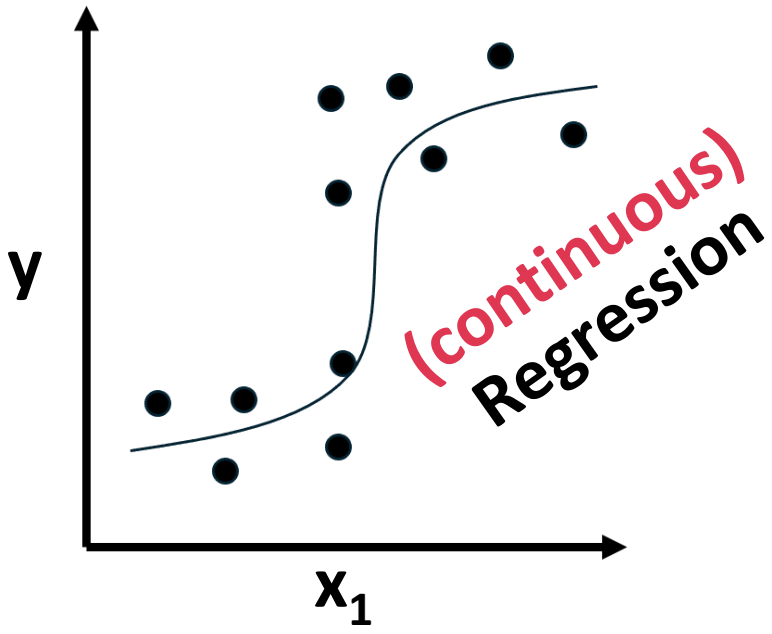
Nature of your “labels” (i.e., predictand or y)

Pixel (i.e., grid cell or point value)
E.g., artificial neural network (dense layers).

Image-based (i.e., spatial region)
E.g., convolutional neural network.

Temporal (i.e., time series)
E.g., recurrent neural network.

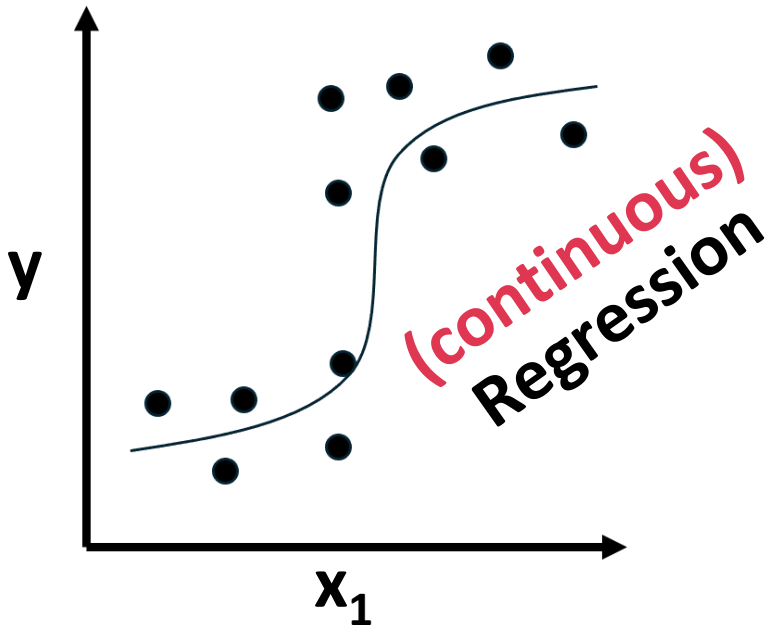
Predict the *value* of something.

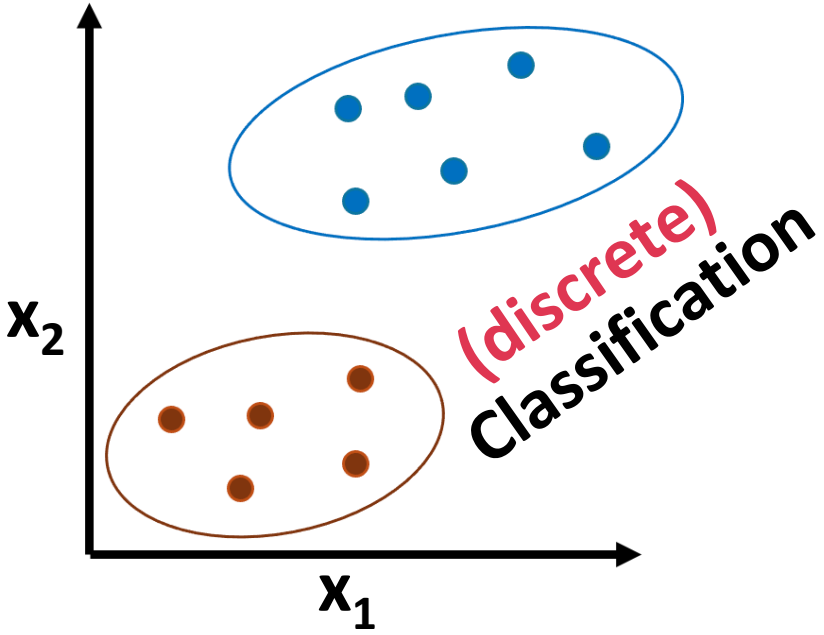


Some examples of regression problems include:

- tropical cyclone maximum wind speed at a certain lead time.
- maximum land surface temperature in an urban area.
- percentage of cloud coverage over a predefined domain.
- bias correction of subseasonal forecasts of precipitation.

Predict the *value* of something.





Predict the *class* of something.

Examples of classification problems:

- whether a thunderstorm will produce severe hazards or be non-severe (i.e., binary).
- whether a polynya is occurring in the Ross Sea (i.e., binary).
- type of cloud(s) that is(are) in a citizen science photograph (i.e., multiclass).
- subseasonal prediction of a North American weather regime class (i.e., multiclass).

Take a guess…

If you had to prioritize one of the areas below for your ML project, which one would have the most impact?

a. Using the latest optimization algorithm

b. Quality and size of your training data

c. Deeper network

d. More clever loss function

Take a guess…

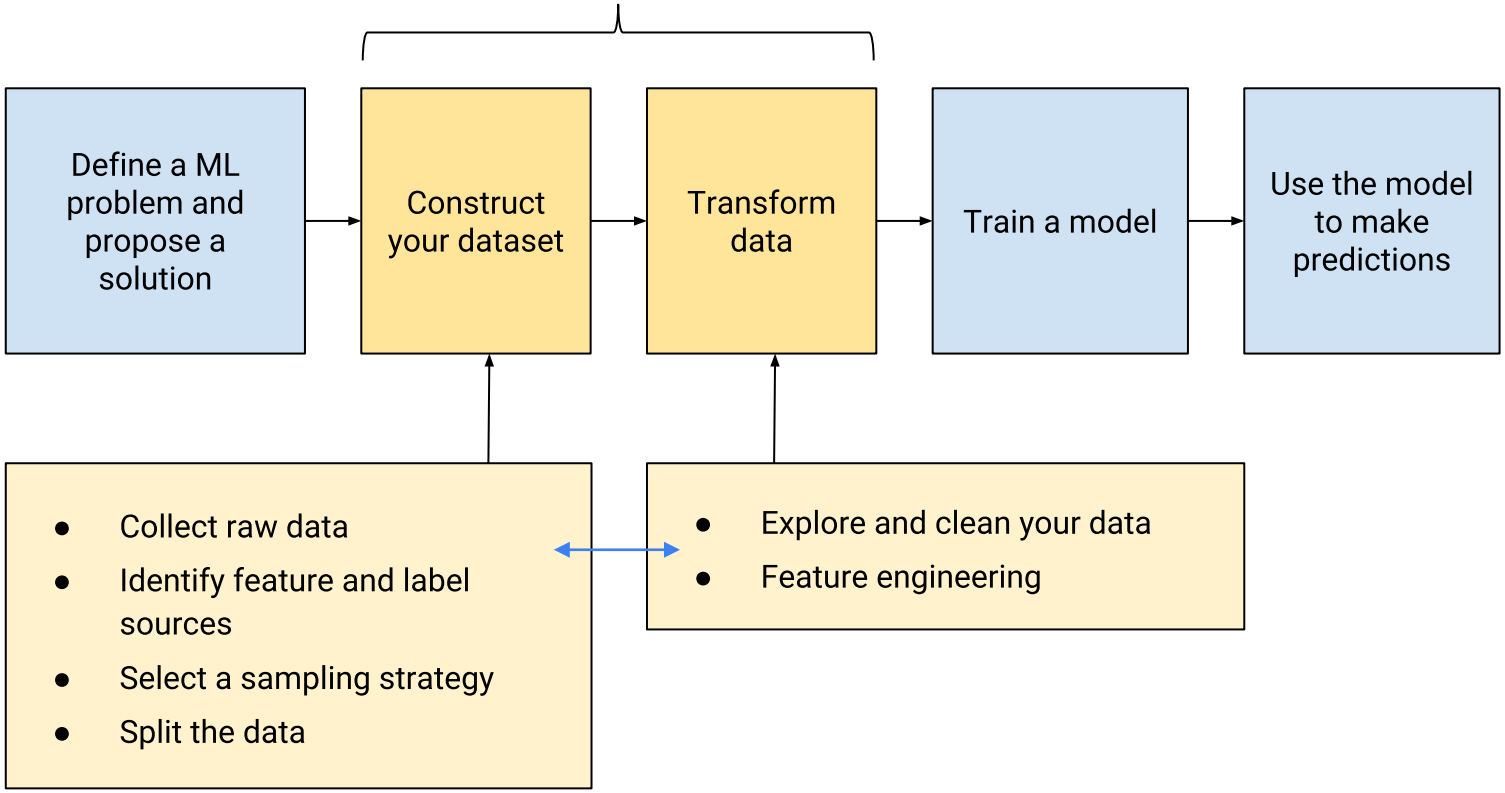
If you had to prioritize one of the areas below for your ML project, which one would have the most impact?

a. Using the latest optimization algorithm

b. Quality and size of your training data

c. Deeper network

d. More clever loss function



https://developers.google.com/machine-learning/data-prep

Questions to ask yourself…

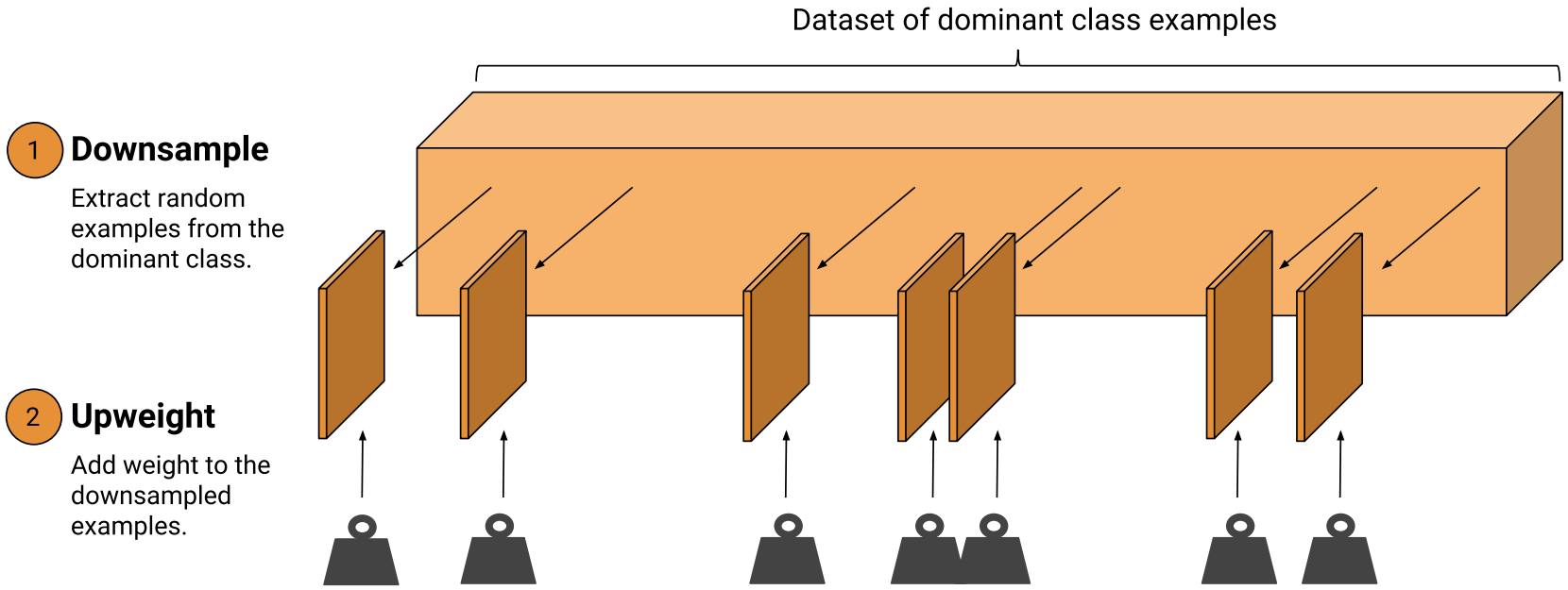
- How many features (i.e., variables) should I use?
- Is my data reliable, representative, or skewed?
- Will I use direct labels or derived (proxy) labels?
- Do I have too much data or imbalanced data?
- How should I split my data into train and test sets?
- How and when should I transform my data?
- How do I encode my classes?

So what can you do about this class issue?

Downsampling and Upweighting

- **Downsampling** (in this context) means training on a disproportionately low subset of the majority class examples.
- **Adding minority class weights** means adding example weight(s) to the minority class.

Molina, M. J., T. A. O'Brien, G. Anderson, M. Ashfaq, K. E. Bennett, W. D. Collins, K. Dagon, J. M. Restrepo, and P. A. Ullrich (2023). A Review of Recent and Emerging Machine Learning Applications for Climate Variability and Weather Phenomena. Artificial Intelligence for the Earth Systems.



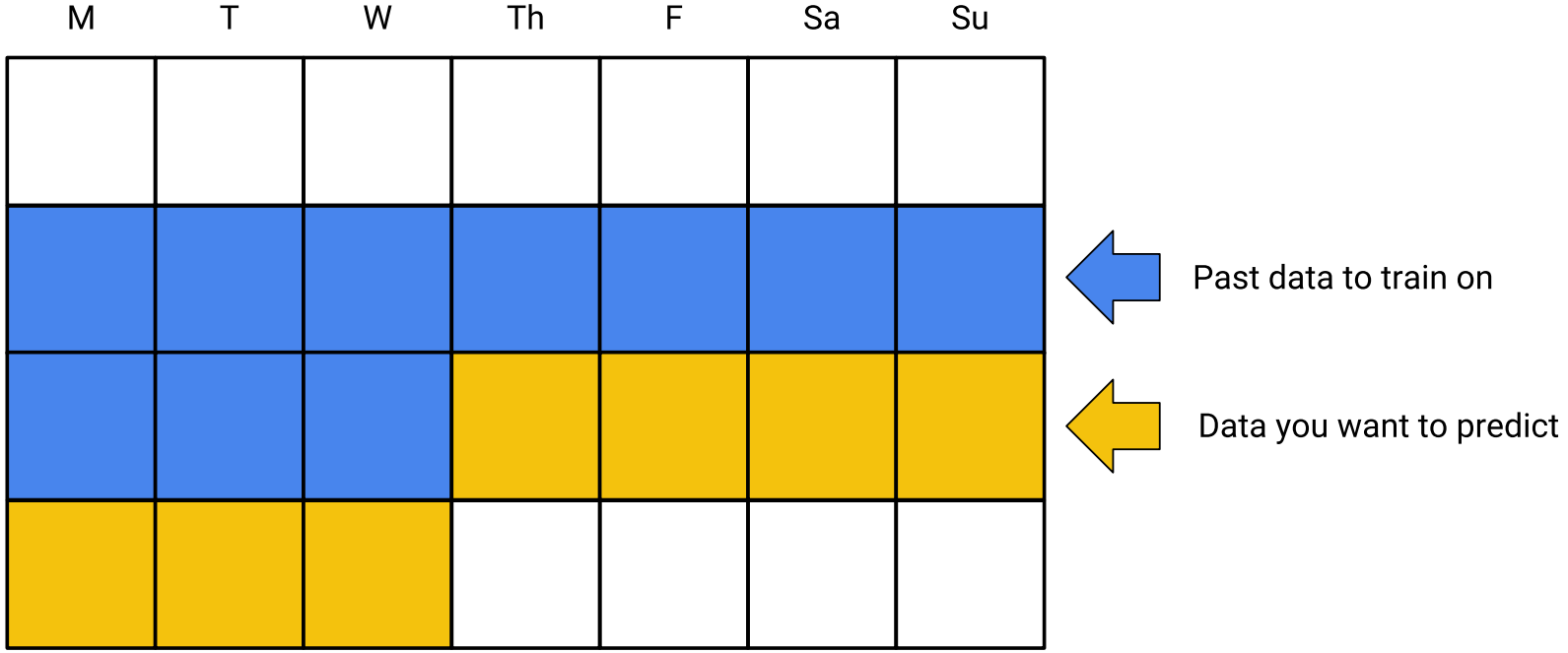
$$\{\text{example weight}\} = \{\text{original example weight}\} \times \{\text{downsampling factor}\}$$

Faster convergence: During training, we see the minority class more often, which will helps converge faster.
Disk space: Consolidating the majority class into fewer examples with larger weights, we spend less disk space storing them.
Calibration: Upweighting ensures our model is still calibrated; the outputs can still be interpreted as probabilities.

https://developers.google.com/machine-learning/data-prep

Seasonality, trends, or cyclical effects can be an issue!

Consider removing these effect during preprocessing or using the date as a feature so the model can learn these effects.



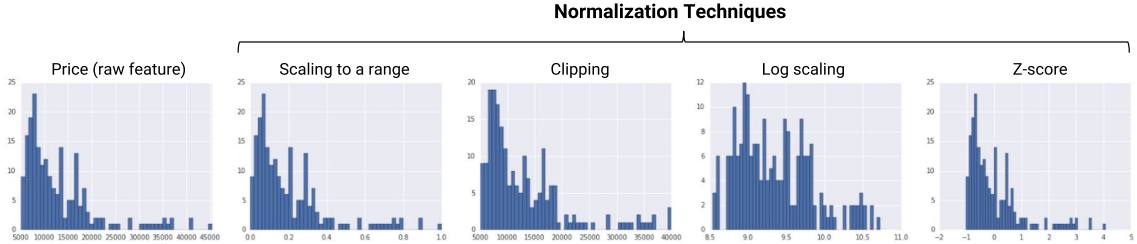
https://developers.google.com/machine-learning/data-prep

Where/when to do transformations?

Transformations can be done before or during training.

Good when data statistics are needed. Dangerous during inference. Slow when iterating.

Easy and fast to iterate. Transforms per batch can slow training if complicated.

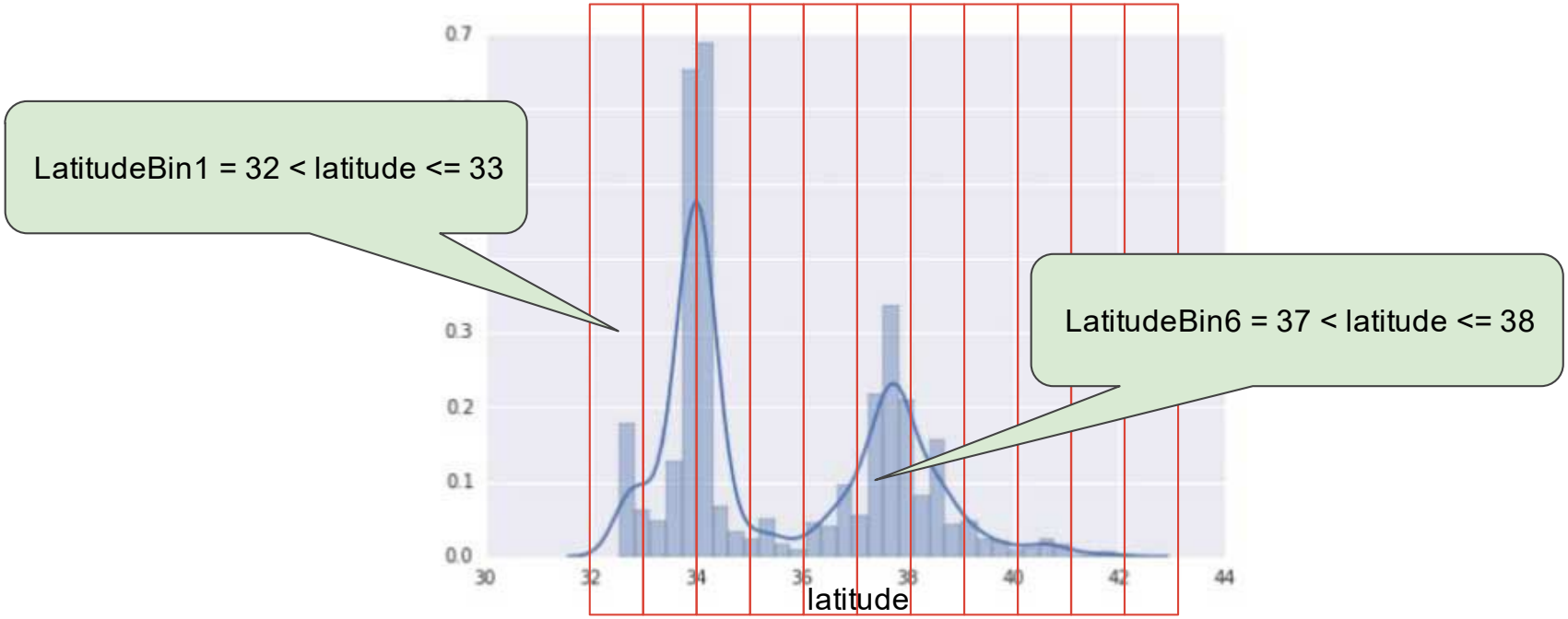


https://developers.google.com/machine-learning/data-prep

We can frame our problem to be **binary** classification (two classes; e.g., True or False, or Yes or No problems).

Classification problems can also be **multiclass** in nature.

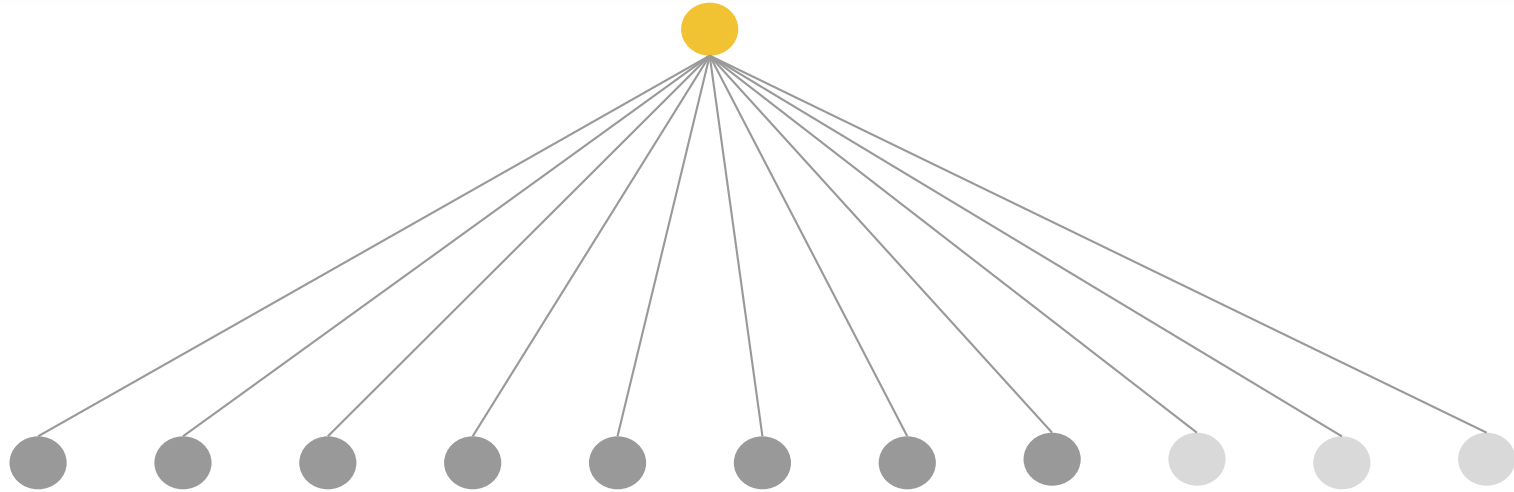
Continuous data could be grouped or binned into certain classes (i.e., buckets); *continuous data can be converted to classes*.



https://developers.google.com/machine-learning/data-prep

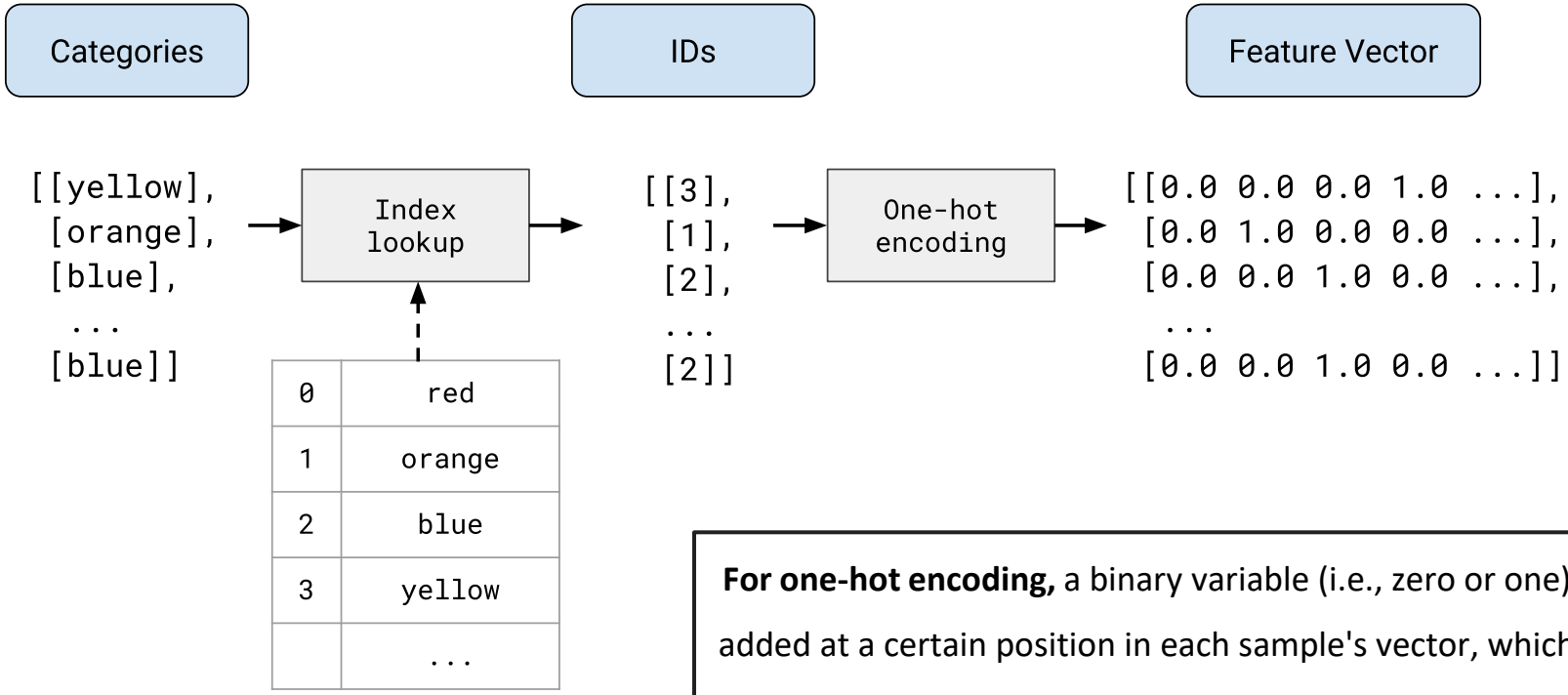
An example of a common occurrence of binning:

- Bin continuous maximum tropical cyclone wind speeds into hurricane intensities based on the Saffir-Simpson Hurricane Wind Scale, e.g., category 1, category 2, etc.



Integer encoding involves assigning an integer value to each unique category. This can work well (not mandatory) for data that have an underlying natural ordered relationship between each other, since machine learning models may be able to understand and harness this relationship.

https://developers.google.com/machine-learning/data-prep

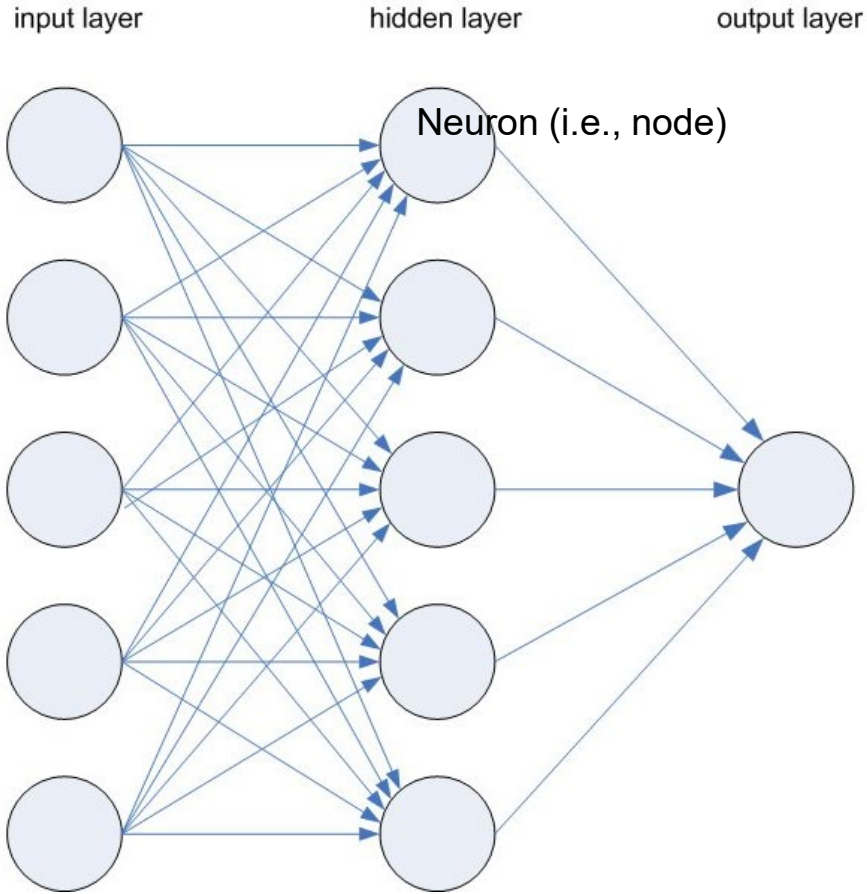


For one-hot encoding, a binary variable (i.e., zero or one) is added at a certain position in each sample's vector, which is meant to represent the respective sample's class, e.g., [0, 0, 1] vs [1, 0, 0].

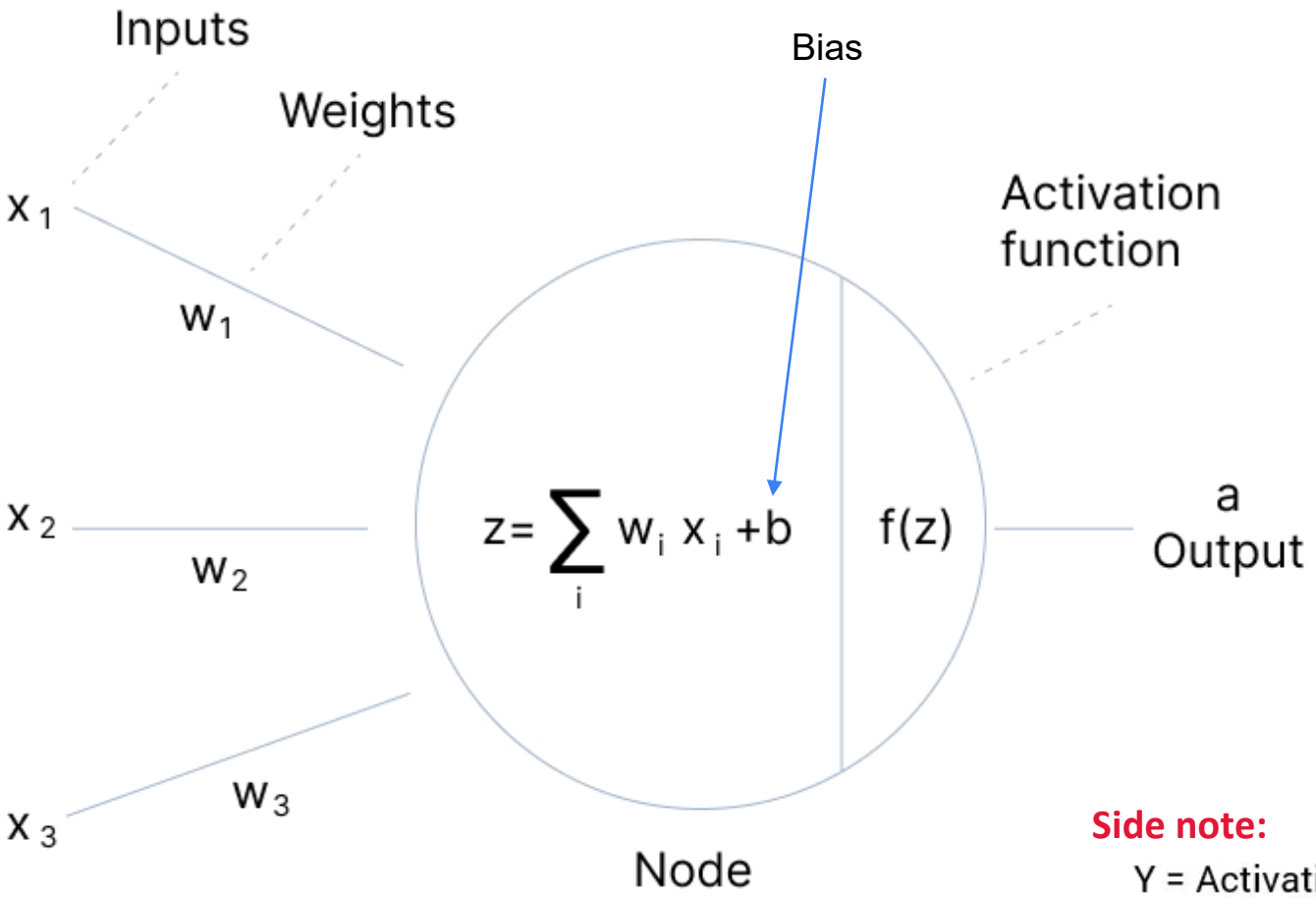
https://developers.google.com/machine-learning/data-prep

Neural networks are machine learning models consisting of layers and neurons (i.e., nodes).

The basic units are neurons (i.e., nodes), which are typically organized into layers.



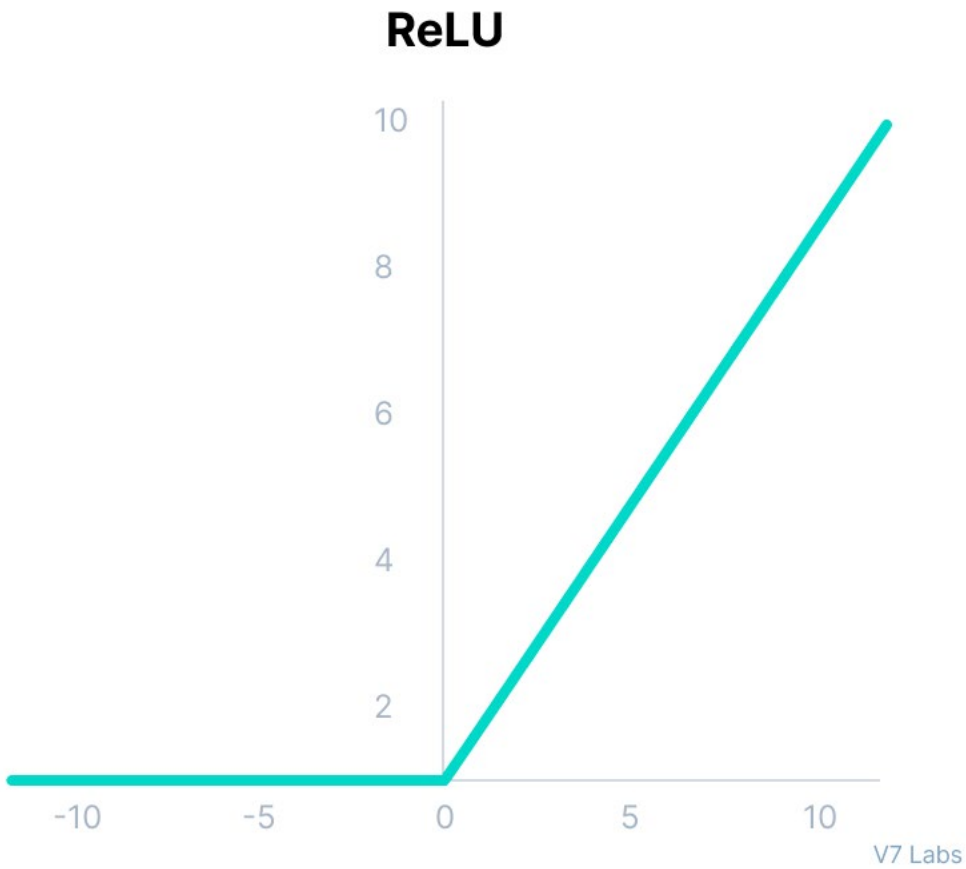
https://www.ibm.com/docs/en/spss-modeler/18.0.0?topic=networks-neural-model



Weights and bias terms are learned. Activation function is not; it is a "hyperparameter."

Side note:

Y = Activation($\Sigma(weight * input) + bias$)



$$f(x) = max\,(0, x)$$

Since ReLU outputs zero for all negative inputs, it naturally leads to sparse activations; at any time, only a subset of neurons are activated, leading to more efficient computation.

https://www.v7labs.com/blog/neural-networks-activation-functions

Hyperparameters are parameters whose values are set before starting the model training process.

Method	Description
Learning rate	Determines the rate update/step size for weights during backpropagation. Usually between 0.01 to 0.0001.
Activation functions	Function that calculates the output of a node based on the sum of the inputs times weights and bias. Can add nonlinearity.
Weight initialization	Method to initialize the weights. Can be zero, one, random…
Preprocessing choices	How you preprocess your data is an important hyperparameter!
Loss function	What error would you like to minimize? E.g., MSE, RMSE, etc.

Hyperparameters are parameters whose values are set before starting the model training process.

Method	Description
Epoch	How many times the entire training dataset can be “seen” by the neural network. Too many epochs can lead to overfitting.
Batch size	How many samples from the training set can be “seen” at one time by the neural network to compute error with and perform backprop. Usually numbers like 16, 32, 64, 128, or 256.
Train and test split	How you split your dataset can impact neural network performance. 80/20 or 60/40 splits are common.
Layer types	Types of layers (e.g., dense, fully connected).

That’s it for me!

Over to Tom Beucler now!

3) How are ML models trained & evaluated?

4) What is physics-guided ML?

That’s it for me!

Over to Tom Beucler now!

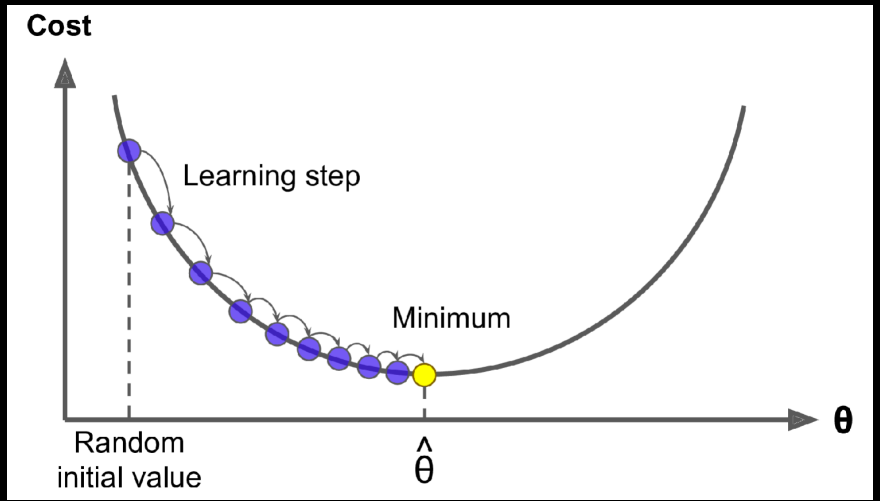
3) How are ML models trained & evaluated?

4) What is physics-guided ML?

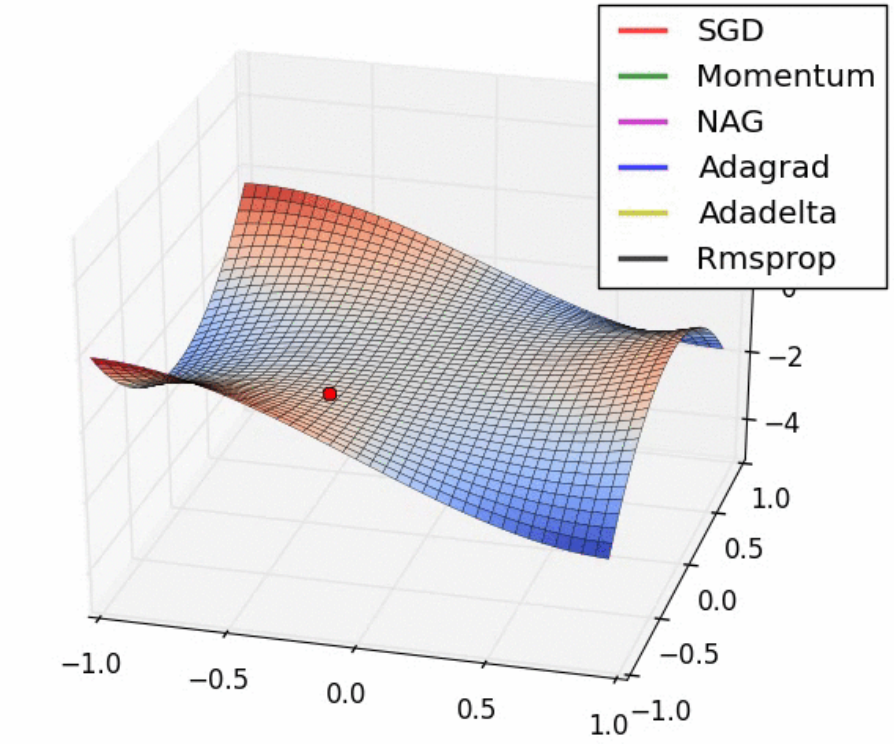
1) What is a cost/loss/error function?

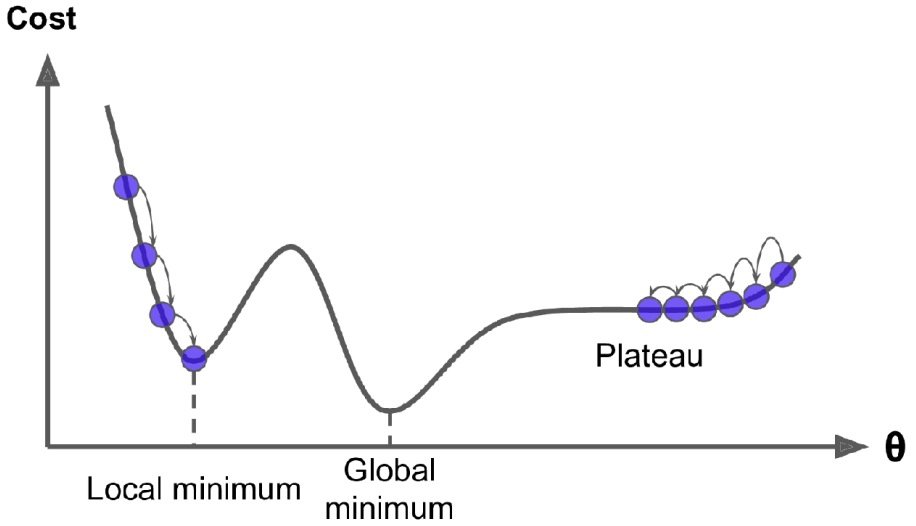
Function: (Predictions, [Truth], [Inputs]) $\longmapsto$ (Cost/Loss/Error to minimize)

For complex data, usually minimized using gradient descent:



1) Training ML Models: Gradient Descent





Batch vs Stochastic vs Mini-Batch Gradient Descent

Source: Computational Optimization (Cornell University)

2) Many options to train (and evaluate) classifiers!

Classification metrics

See the Classification metrics section of the user guide for further details.

metrics.accuracy_score(y_true, y_pred, *[, ...])	Accuracy classification score.
metrics.auc(x, y)	Compute Area Under the Curve (AUC) using the trapezoidal rule.
metrics.average_precision_score(y_true, ...)	Compute average precision (AP) from prediction scores.
metrics.balanced_accuracy_score(y_true, ...)	Compute the balanced accuracy.
metrics.brier_score_loss(y_true, y_prob, *)	Compute the Brier score loss.
metrics.classification_report(y_true, y_pred, *)	Build a text report showing the main classification metrics.
metrics.cohen_kappa_score(y1, y2, *[, ...])	Cohen's kappa: a statistic that measures inter-annotator agreement.
metrics.confusion_matrix(y_true, y_pred, *)	Compute confusion matrix to evaluate the accuracy of a classification.
metrics.dcg_score(y_true, y_score, *[, k, ...])	Compute Discounted Cumulative Gain.
metrics.det_curve(y_true, y_score[, ...])	Compute error rates for different probability thresholds.
metrics.f1_score(y_true, y_pred, *[, ...])	Compute the F1 score, also known as balanced F-score or F-measure.
metrics.fbeta_score(y_true, y_pred, *, beta)	Compute the F-beta score.
metrics.hamming_loss(y_true, y_pred, *[, ...])	Compute the average Hamming loss.
metrics.hinge_loss(y_true, pred_decision, *)	Average hinge loss (non-regularized).
metrics.jaccard_score(y_true, y_pred, *[, ...])	Jaccard similarity coefficient score.
metrics.log_loss(y_true, y_pred, *[, eps, ...])	Log loss, aka logistic loss or cross-entropy loss.
metrics.matthews_corrcoef(y_true, y_pred, *)	Compute the Matthews correlation coefficient (MCC).
metrics.multilabel_confusion_matrix(y_true, ...)	Compute a confusion matrix for each class or sample.
metrics.ndcg_score(y_true, y_score, *[, k, ...])	Compute Normalized Discounted Cumulative Gain.
metrics.precision_recall_curve(y_true, ...)	Compute precision-recall pairs for different probability thresholds.
metrics.precision_recall_fscore_support(...)	Compute precision, recall, F-measure and support for each class.
metrics.precision_score(y_true, y_pred, *[, ...])	Compute the precision.
metrics.recall_score(y_true, y_pred, *[, ...])	Compute the recall.
metrics.roc_auc_score(y_true, y_score, *[, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
metrics.roc_curve(y_true, y_score, *[, ...])	Compute Receiver operating characteristic (ROC).
metrics.top_k_accuracy_score(y_true, y_score, *)	Top-k Accuracy classification score.
metrics.zero_one_loss(y_true, y_pred, *[, ...])	Zero-one classification loss.

Source: scikit-learn

2) Confusion Matrix of the task: **Is the digit 5?**

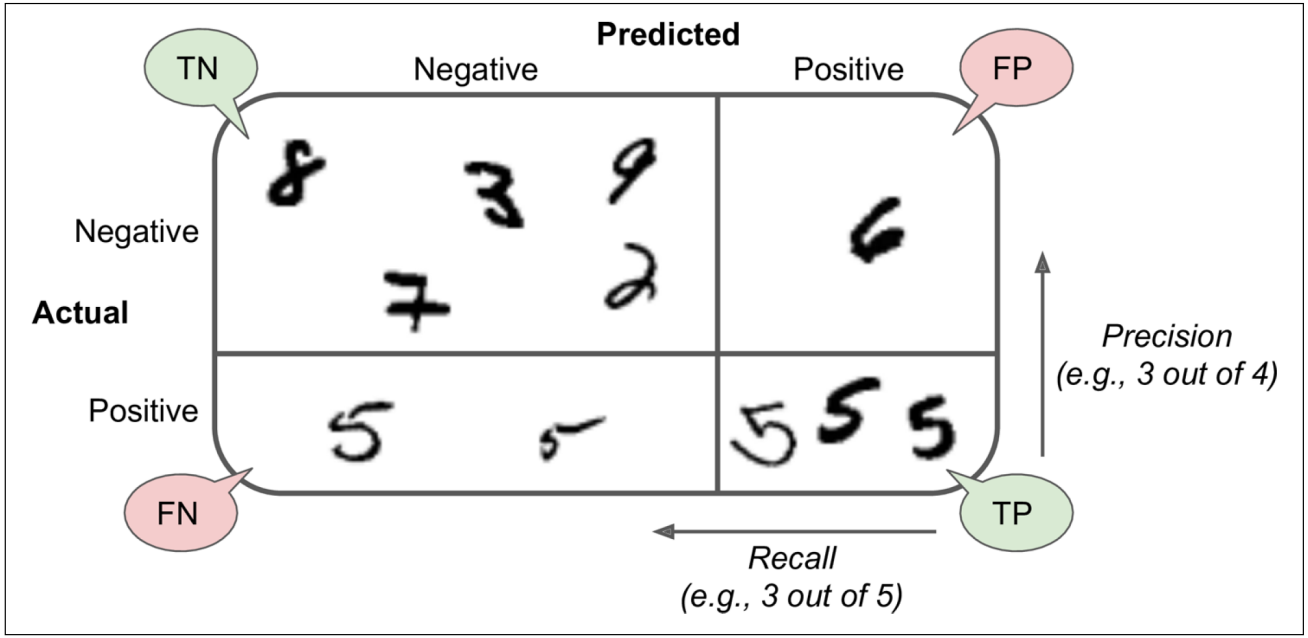
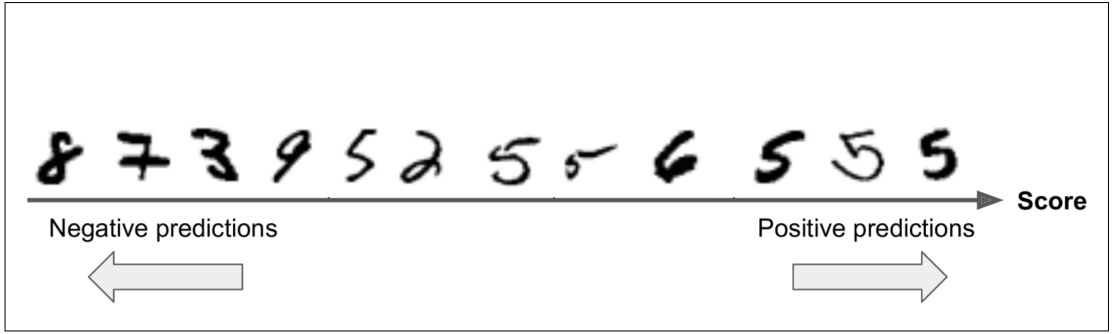


Figure 3-2. An illustrated confusion matrix shows examples of true negatives (top left), false positives (top right), false negatives (lower left), and true positives (lower right)

Accuracy:

$$\frac{TP+TN}{TP+TN+FP+FN}$$



$$\text{recall} = \frac{TP}{TP + FN}$$

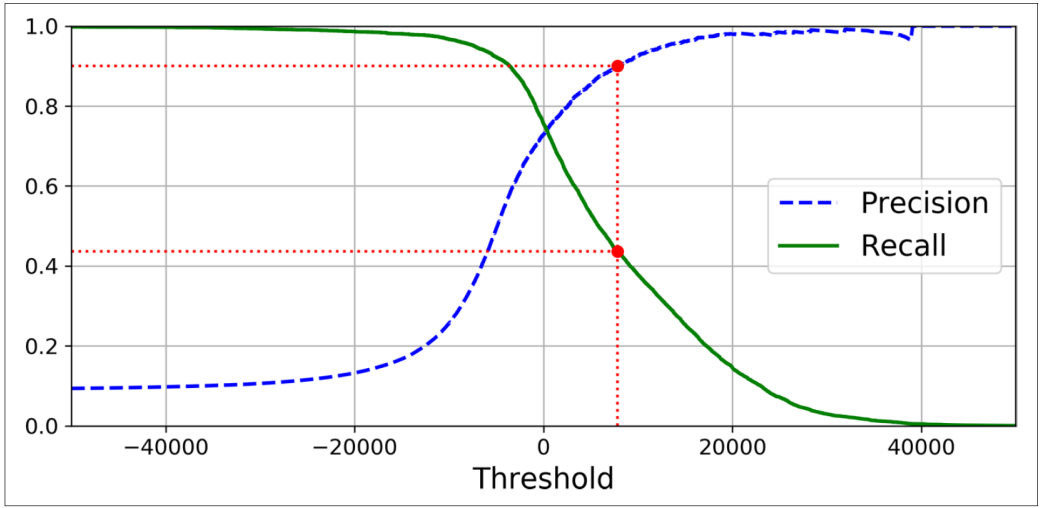


Figure 3-4. Precision and recall versus the decision threshold

$$\text{precision} = \frac{TP}{TP + FP}$$

3) Loss/Cost/Error Functions for Regression Models

Mean Squared Error:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

“Truth” Pred.

For environmental applications:

- Root-Mean-Square Error (RMSE)
- Mean Absolute Error

Are popular because they preserve units!

3) Many options to train (and evaluate) regression models!

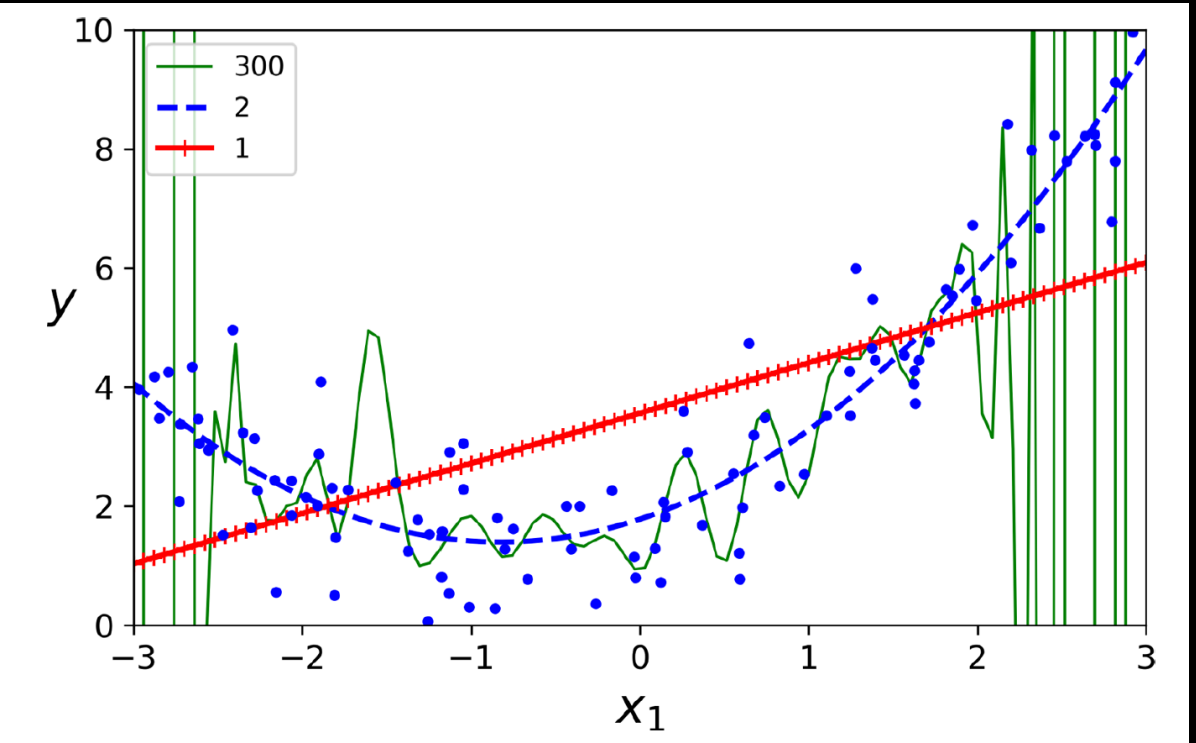
Regression metrics

See the Regression metrics section of the user guide for further details.

`metrics.explained_variance_score`(y_true, ...)	Explained variance regression score function.
`metrics.max_error`(y_true, y_pred)	The max_error metric calculates the maximum residual error.
`metrics.mean_absolute_error`(y_true, y_pred, *)	Mean absolute error regression loss.
`metrics.mean_squared_error`(y_true, y_pred, *)	Mean squared error regression loss.
`metrics.mean_squared_log_error`(y_true, y_pred, *)	Mean squared logarithmic error regression loss.
`metrics.median_absolute_error`(y_true, y_pred, *)	Median absolute error regression loss.
`metrics.mean_absolute_percentage_error`(...)	Mean absolute percentage error (MAPE) regression loss.
`metrics.r2_score`(y_true, y_pred, *[, ...])	R^2 (coefficient of determination) regression score function.
`metrics.mean_poisson_deviance`(y_true, y_pred, *)	Mean Poisson deviance regression loss.
`metrics.mean_gamma_deviance`(y_true, y_pred, *)	Mean Gamma deviance regression loss.
`metrics.mean_tweedie_deviance`(y_true, y_pred, *)	Mean Tweedie deviance regression loss.
`metrics.d2_tweedie_score`(y_true, y_pred, *)	D^2 regression score function, percentage of Tweedie deviance explained.
`metrics.mean_pinball_loss`(y_true, y_pred, *)	Pinball loss for quantile regression.

Source: scikit-learn

How can ML best practices help us consistently assess the performance of our three models?

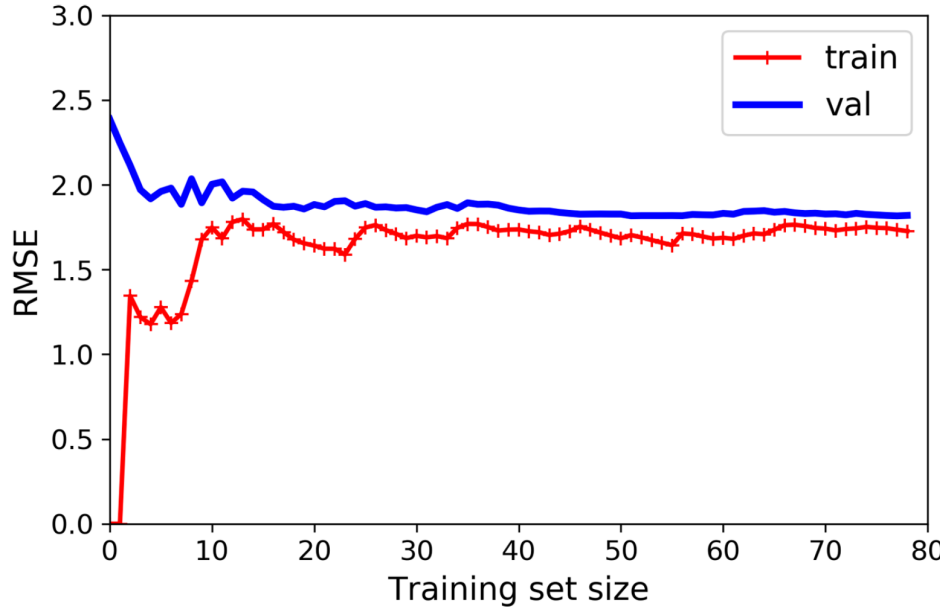


Which model would you choose in practice? Why?

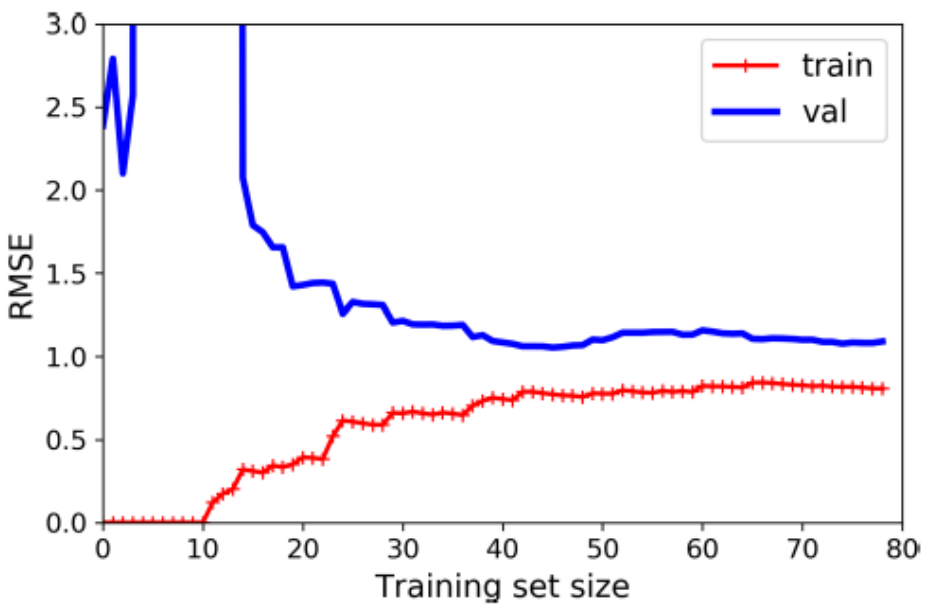
The quadratic model because the polynomial regression model is overfitting the training data and the linear model is underfitting the training data.

Cross-Validation (& Learning Curves) Help Assess how ML Models Generalize to Out-of-Sample Data

Linear Regression: Underfitting



High-degree Polynomial: Overfitting



How to benchmark a regression model?

Step 1: Pick performance metrics (MSE, RMSE, MAE, etc.)

Coefficient of **determination** (between -∞ and 1):

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

← Sum of squared errors (proportional to MSE)

← Total Sum of squares (proportional to true variance)

≠ Coefficient of **correlation:** (between -1 and 1)

$$\frac{\text{cov}(Y_{true}, Y_{pred})}{\text{std}(Y_{true}) \times \text{std}(Y_{pred})}$$

How to benchmark a regression model?

Step 2: Report mean performance over training, validation, and test sets

	Training Set	Validation Set	Test Set
RMSE (kg)	10	11	50
Coefficient of Determination	0.6	0.45	-2

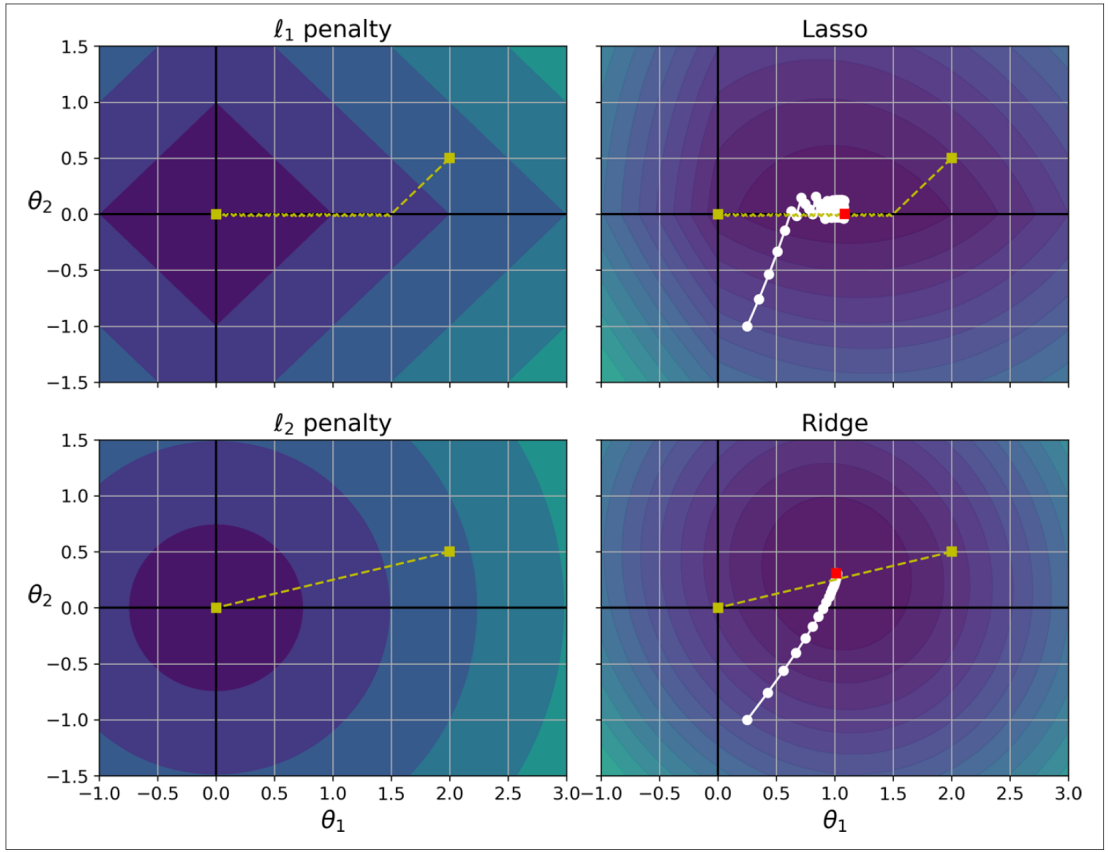


Figure 4-19. Lasso versus Ridge regularization

Lasso: Penalizes the absolute value of the weights (L1 norm)

≠

Ridge: Penalizes the squares of the weights (L2 norm)

Equation 4-10. Lasso Regression cost function

$$J(\mathbf{\theta}) = \mathrm{MSE}(\mathbf{\theta}) + \alpha \Sigma_{i=1}^{n} |\theta_i|$$

Performance

Regularization

That’s it for me!

Over to Tom Beucler now!

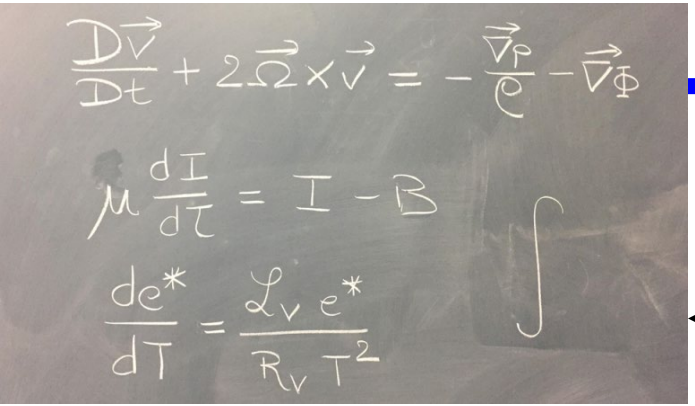
3) How are ML models trained & evaluated?

4) What is physics-guided ML?

Physics-Guided ML: Add physical structure to restrict ML output to physically-plausible solutions

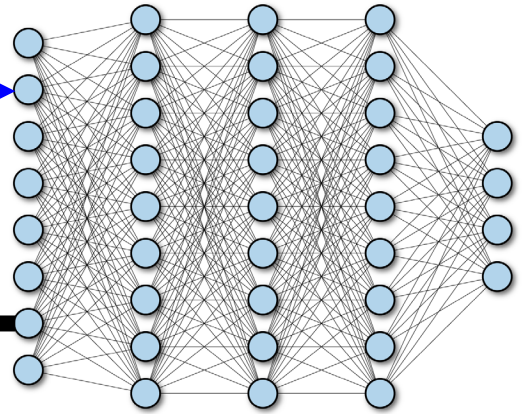
$$\text{Relative Humidity} \overset{\text{def}}{=} \frac{\text{Vapor pressure}}{\text{Saturation vapor pressure}}$$

Physics-Guided ML: Add physical structure to restrict ML output to physically-plausible solutions



Incorporate physical knowledge into ML

Physics-Guided ML



Data-Driven Discovery

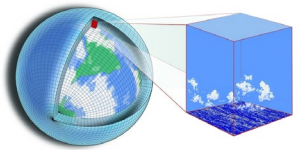
Use ML to extract physical knowledge from data

Reviews: Willard et al. (2020), Kashinath et al. (2021), Reichstein et al. (2019), Karpatne et al. (2017), Beucler et al. (2021)

TROPICANA Physics-Guided ML Tutorial

Advantages	Limitations
• Physical consistency *small violations physical def./laws* • Robustness (control) *stability to small perturbations* • Improved Generalization *extrapolation to regimes/extremes* • Data efficiency *ability to learn with few samples* • Interpretability *explainable structure & behavior* • Reduced error*	• Only a few laws hold exactly: *definitions, state eq., cons. laws* • Sources and sinks may be difficult to quantify in real data *importance of uncertainty quant.* • Trade-off between physical consistency and optimization *"no free lunch" especially when dim(Constraints) << effective dim.*

Physics-Guided ML: Add physical structure to restrict ML output to physically-plausible solutions

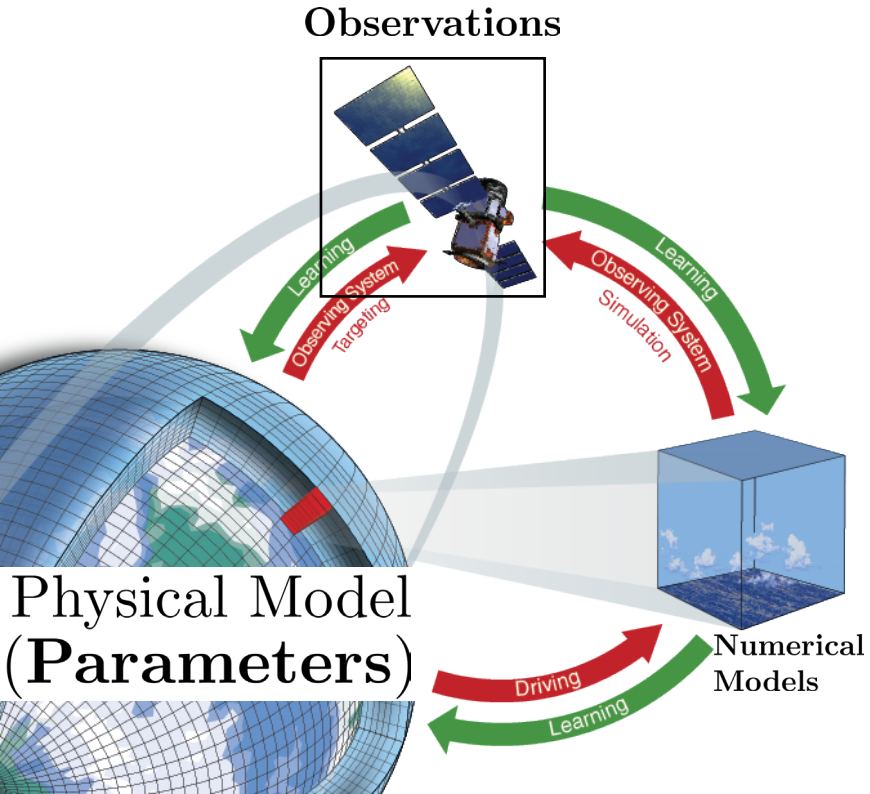


Physical Structure

Learn Parameters of Physical Model

See: Lopez-Gomez et al. (2020) Schneider et al. (2017), Reichstein et al. (2019), Camps-Vall et al. (2018), Image Source: CliMA

(Most constrained) Learn Parameters of Physical Models



Parameter estimation/calibration problem

Subfield of data assimilation = Optimal state estimation given real-time data

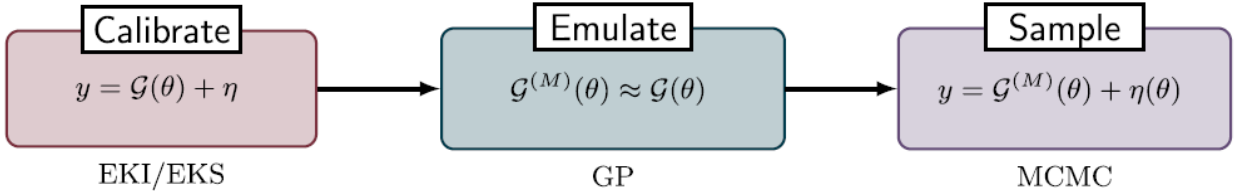
See ECMWF resources on 4D-Var

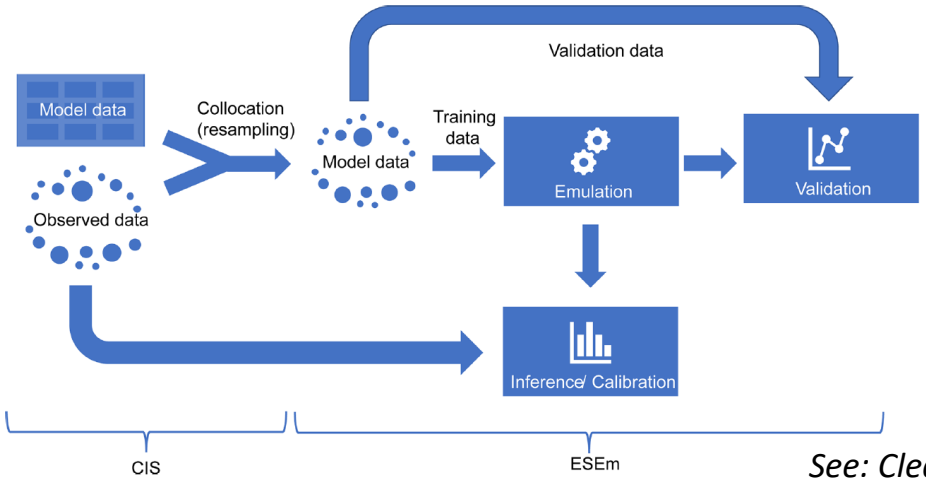
$$p\,(\text{Parameters}|\text{Obs})$$

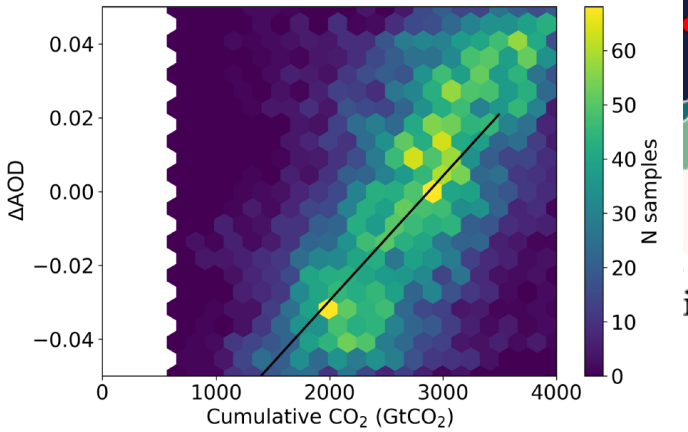
ML helps maximize the likelihood

See: Brajard et al. (2021), ECMWF Fact sheets; Image Source: CliMA

(Most constrained) Learn Parameters of Physical Models
ML-based frameworks: CES & ESEm

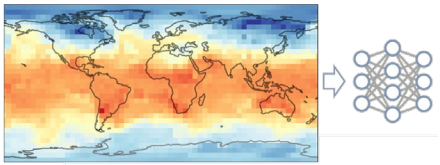


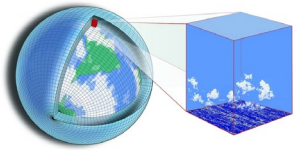




See: Cleary et al. (2021), Howland et al. (2022), Watson-Parris et al. (2021)

Physics-Guided ML: Add physical structure to restrict ML output to physically-plausible solutions





Physical Structure

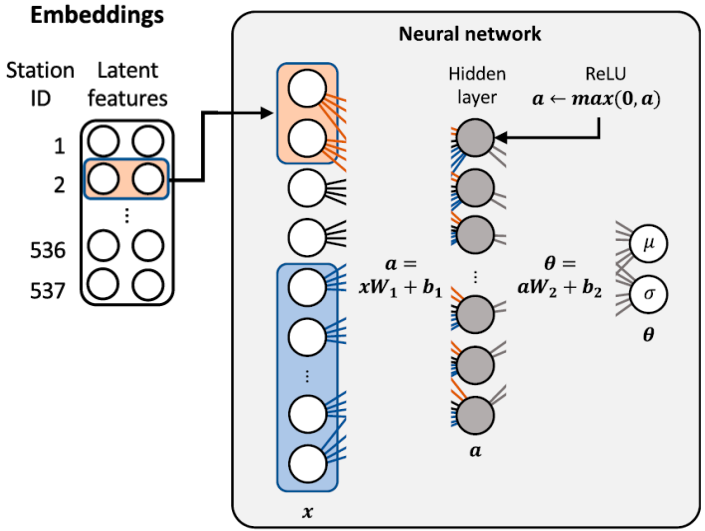
Bias Correction of Physical Model

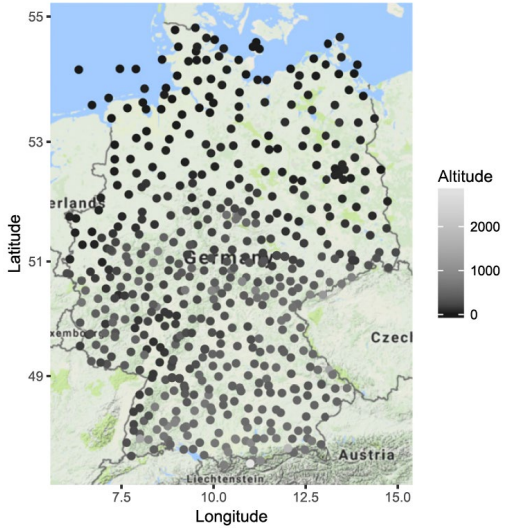
Learn Parameters of Physical Model

See: Rasp and Lerch (2018), Grönquist et al. (2021), Bonavita and Laloyaux (2020), Image Source: Rasp et al. (2020)

(Usually constrained) Bias-correcting physical model

- Residual modeling: Debiased Model = (Model to correct) + **ML**
  (1) Post-processing (Model Output Statistics)

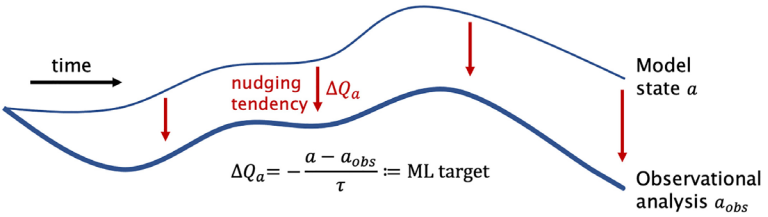


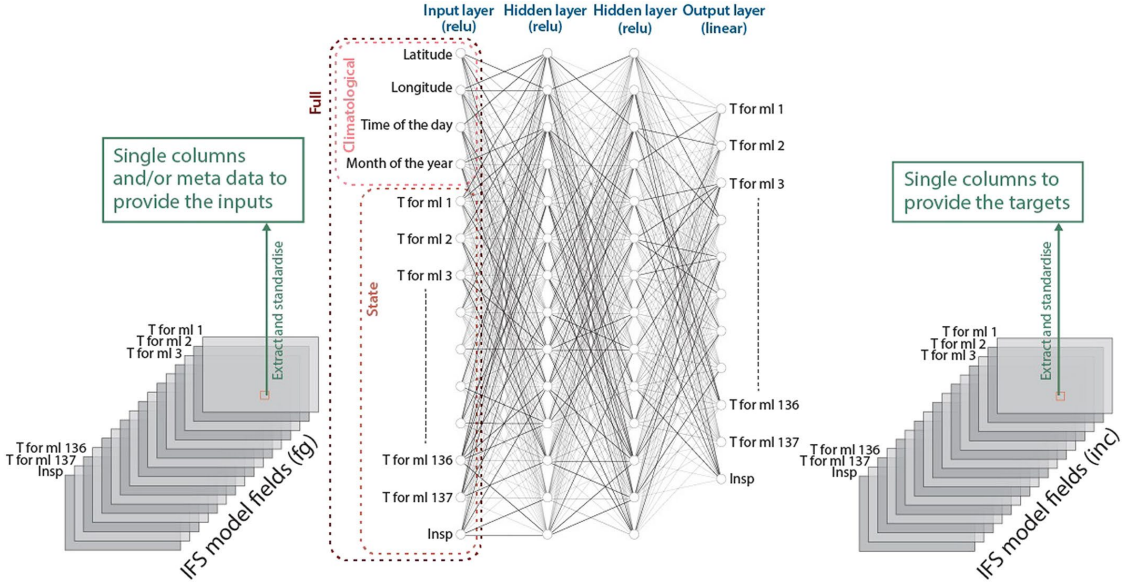


See: Rasp and Lerch (2018), other ML post-processing references

(Usually constrained) Bias-correcting physical model

- Residual modeling:
  (1) Post-processing
  (2) Nudging/DA increments



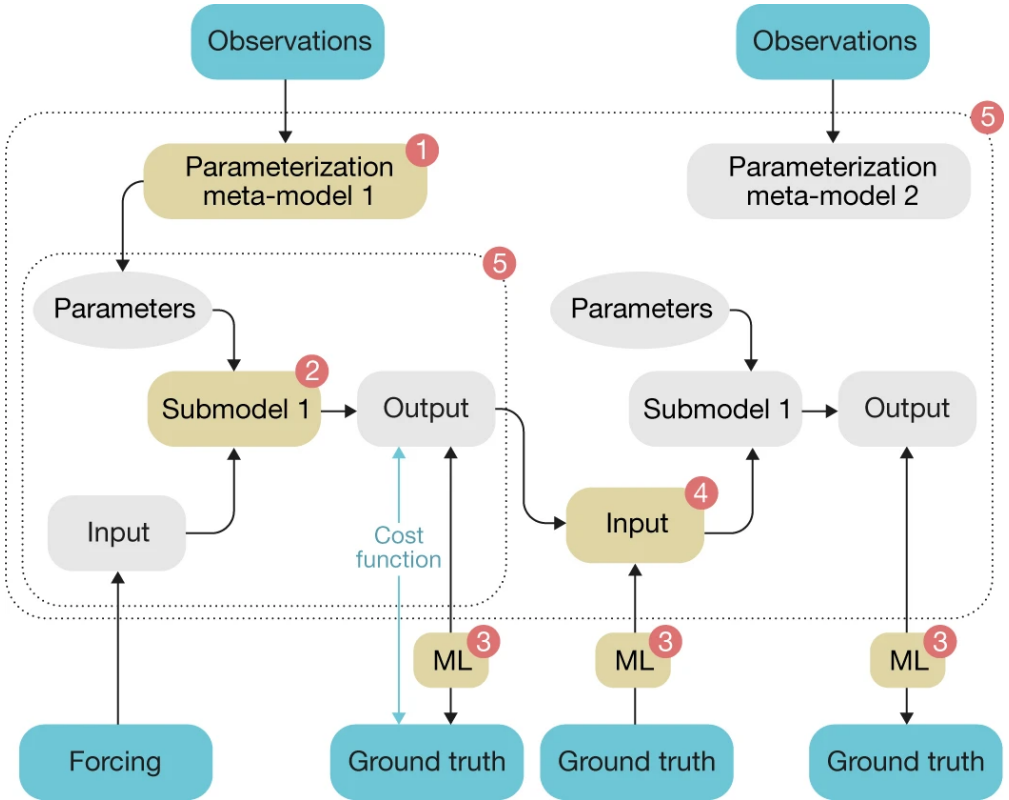


See: Watt-Meyer et al. (2021), Clark et al. (2022), Bonavita and Laloyaux (2020)

(Usually constrained) Bias-correcting physical model

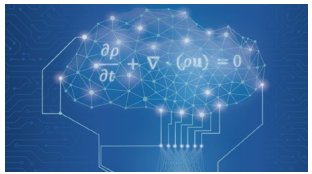
- Residual modeling:
  (1) Post-processing
  (2) Nudging /DA increments

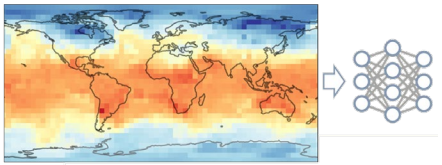
- Hybrid modeling: Combining physical model & ML

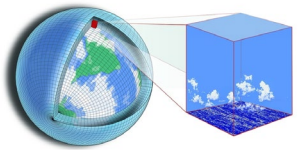


Source: Reichstein et al. (2019)

Physics-Guided ML: Add physical structure to restrict ML output to physically-plausible solutions







Physical Structure

Physics-Constrained Loss or Architecture

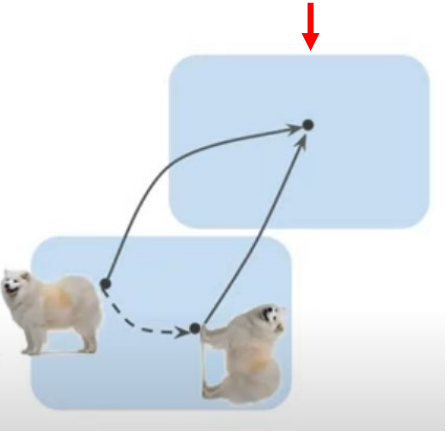
Bias Correction of Physical Model

Learn Parameters of Physical Model

See: Karpatne et al. (2017), Wu et al. (2020), Raissi et al. (2019), Beucler et al. (2021) Image: R. Gauthier-Butterfield, UCI (2021)

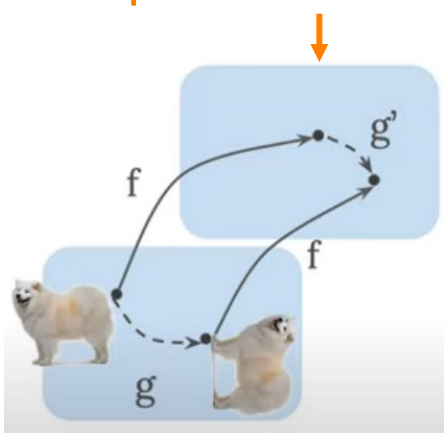
(Hard constraint) Physics-constrained architecture

- Encode invariance or equivariance in architecture



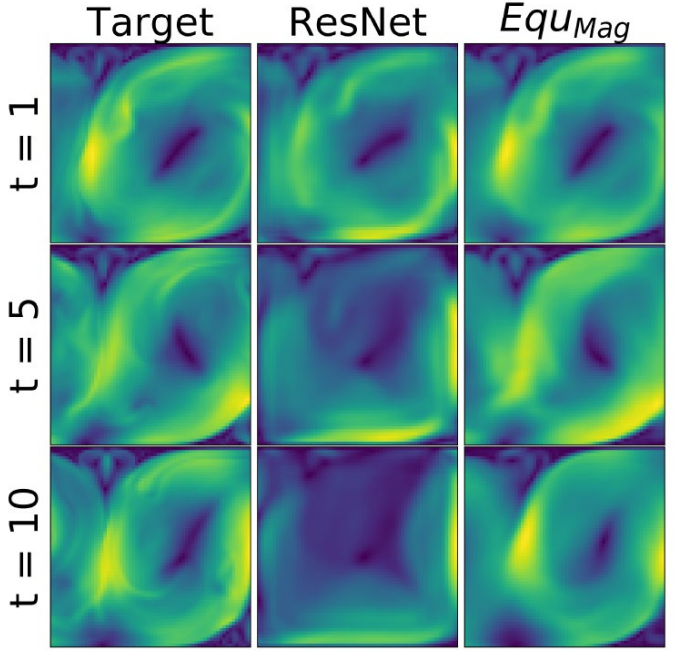
$$f\left[g\left(x\right)\right] = f\left(x\right)$$

Pooling layers
(large pooling size)



$$f\left[g\left(x\right)\right] = g'\left(x\right)$$

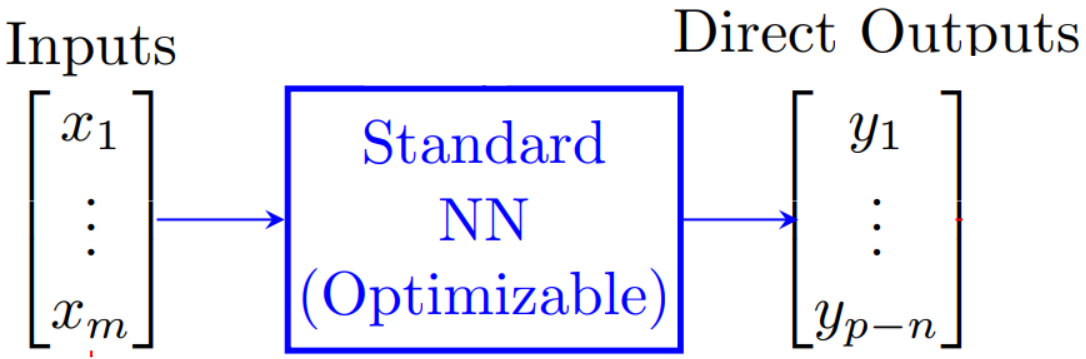
Convolutional
layers



See: Wang et al. (2020), Image adapted from: Kobby Panford-Quainoo (2021)

(Hard constraint) Physics-constrained architecture

- Encode invariance or equivariance in architecture
- Enforce physical constraints via e.g., constraints layers



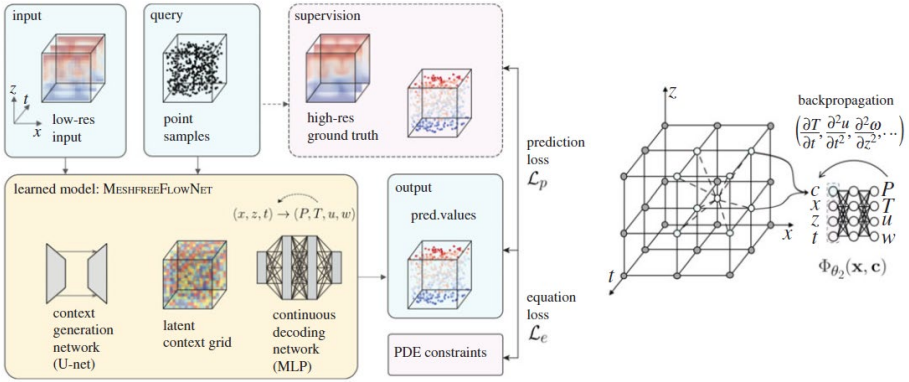
See: Beucler et al. (2021), Zanetta et al. (2022)

(Soft constraint) Physics-constrained loss

$$\text{Loss} = \alpha\,(\text{Regression loss}) + (1-\alpha)\,(\text{Physics} - \text{constrained loss}) \qquad \alpha \in [0,1]$$

Physics-constrained losses: $\left\| \boldsymbol{\mathcal{P}}\left(\boldsymbol{y}_{\mathbf{Pred}}, \boldsymbol{x}\right) \right\|$

- Residual of definitions, state equations, conservation laws



See: Jiang et al. (2020)

(Soft constraint) Physics-constrained loss

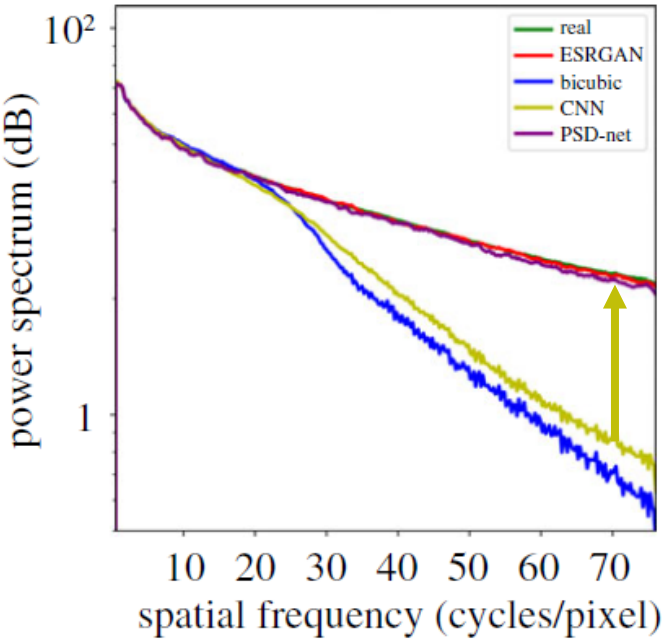
$$\text{Loss} = \alpha\,(\text{Regression loss}) + (1-\alpha)\,(\text{Physics} - \text{constrained loss}) \quad \alpha \in [0,1]$$

Physics-constrained losses: $\left\| \widetilde{\boldsymbol{\mathcal{P}}}\left(\boldsymbol{y}_{\mathbf{Pred}}, \boldsymbol{x}\right) \right\|$

- Residual of definitions, state equations, conservation laws
- Approximate dynamics

(Soft constraint) Physics-constrained loss

$$\text{Loss} = \alpha\,(\text{Regression loss}) + (1-\alpha)\,(\text{Physics}-\text{constrained loss}) \quad \alpha \in [0,1]$$



$$\text{Regression loss}\left[\widehat{\boldsymbol{y}_{\mathbf{pred}}}\left(\boldsymbol{k},\omega\right), \widehat{\boldsymbol{y}_{\mathbf{true}}}\left(\boldsymbol{k},\omega\right)\right]$$

e equations, conservation laws

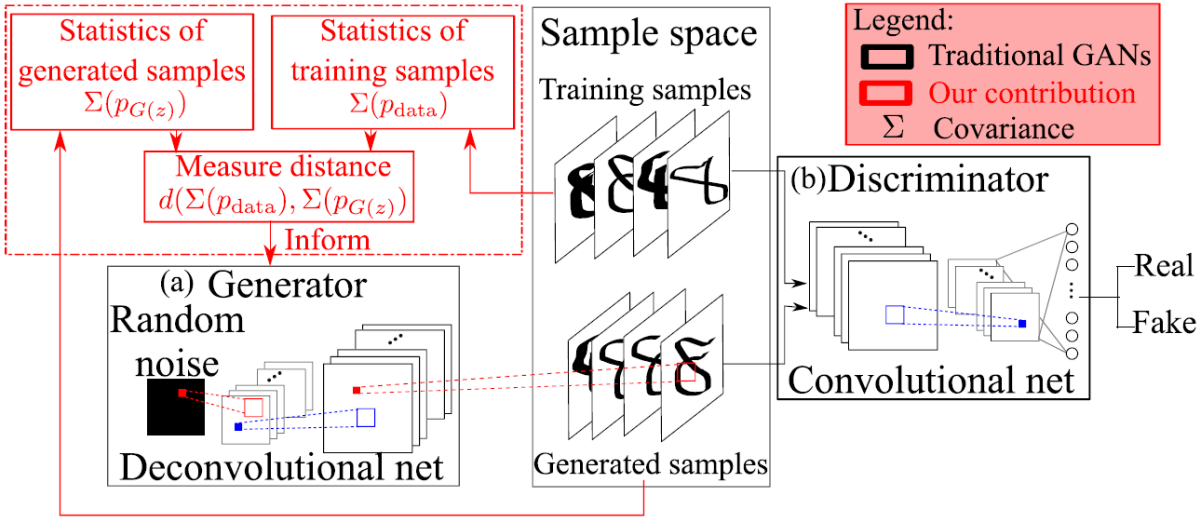
oss; (2) directly predict spectral coefficients

See: Singh et al. (2019), Kashinath et al. (2020)

(Soft constraint) Physics-constrained loss

$$\text{Loss} = \alpha\,(\text{Regression loss}) + (1-\alpha)\,(\text{Physics} - \text{constrained loss}) \quad \alpha \in [0,1]$$

Physics-constrained losses: $\|\mathbf{Covariance}\,(y_{\mathbf{pred}}) - \mathbf{Covariance}\,(y_{\mathbf{true}})\,\|$



ation laws

ict spectral coefficients

oments to truth

See: Wu et al. (2018), Mooers et al. (2020)

(Soft constraint) Physics-constrained loss

$$\text{Loss} = \alpha\,(\text{Regression loss}) + (1-\alpha)\,(\text{Physics} - \text{constrained loss}) \quad \alpha \in [0,1]$$

Physics-constrained losses:

$$\text{Regression loss}\left[\begin{pmatrix}\text{Mean}_{\text{pred}}\\ \text{Std}_{\text{pred}}\\ \text{Skewness}_{\text{pred}}\\ \dots\end{pmatrix}, \begin{pmatrix}\text{Mean}_{\text{true}}\\ \text{Std}_{\text{true}}\\ \text{Skewness}_{\text{true}}\\ \dots\end{pmatrix}\right]$$

- Residual of definitions, state equations, conservation laws
- Approximate dynamics
- (1) Multi-scale or spectral loss; (2) directly predict spectral coefficients
- Statistical constraints: (1) Compare statistical moments to truth
  (2) Predict moments of prescribed distributions

See: Vandal et al. (2017), Vannitsem et al. (2021), Barnes et al. (2021), Haynes et al. (2023)

(Soft constraint) Physics-constrained loss

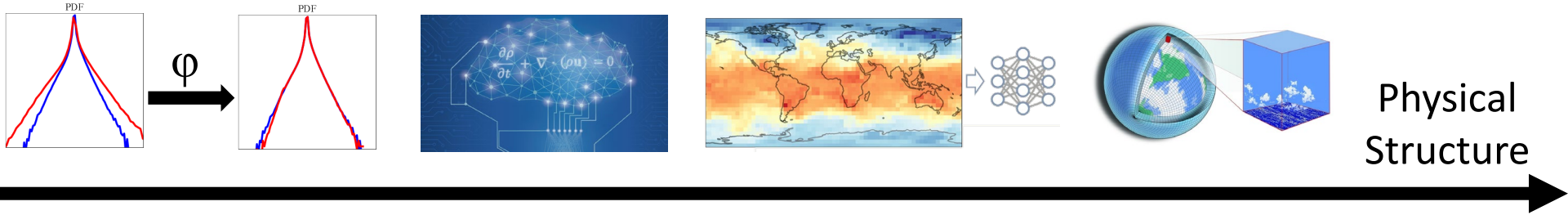
$$\text{Loss} = \alpha\,(\text{Regression loss}) + (1-\alpha)\,(\text{Physics} - \text{constrained loss}) \quad \alpha \in [0,1]$$

Physics-constrained losses: $\text{Regression loss}\,[\boldsymbol{\mathcal{W}}\,(\boldsymbol{y_{\text{pred}}}, \boldsymbol{y_{\text{true}}})]$

- Residual of definitions, state equations, conservation laws
- Approximate dynamics
- (1) Multi-scale or spectral loss; (2) directly predict spectral coefficients
- Statistical constraints: (1) Compare statistical moments to truth
  (2) Predict moments of prescribed distributions
- Custom weighting to emphasize certain physical regimes

See: Lopez-Gomez et al. (2022)

Physics-Guided ML: Add physical structure to restrict ML output to physically-plausible solutions



Physics-Informed Feature selection /transformation

Physics-Constrained Loss or Architecture

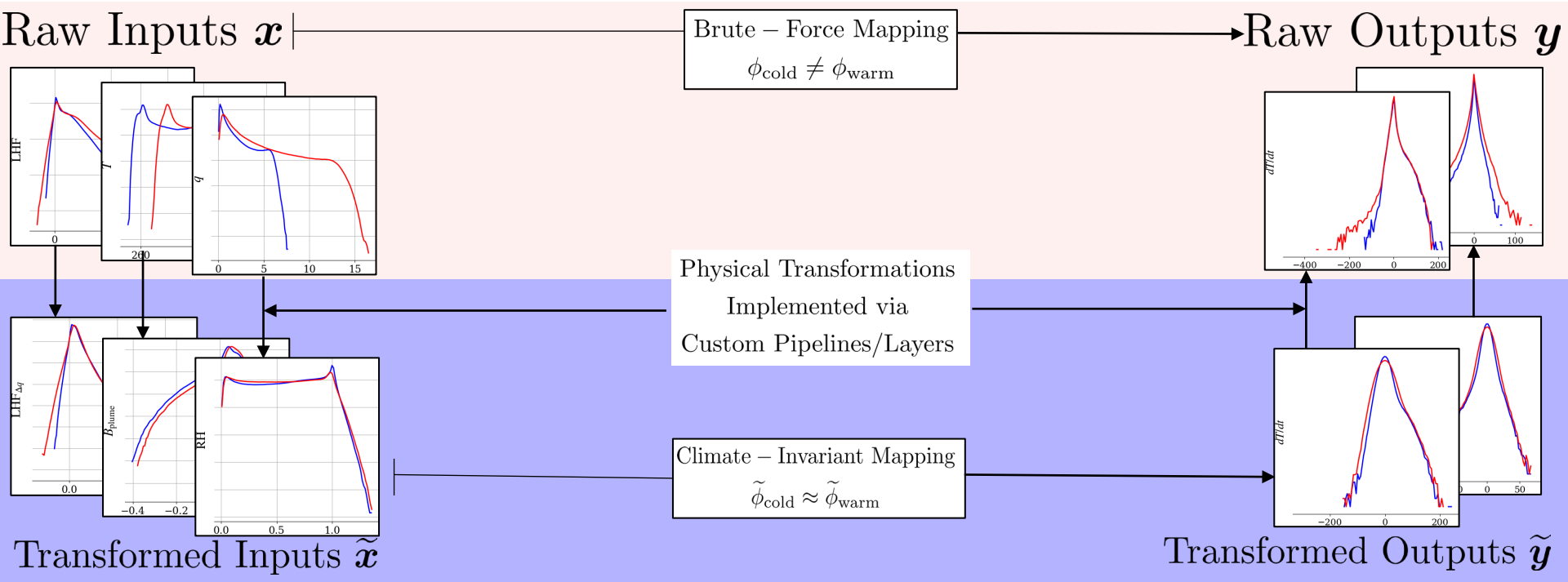
Bias Correction of Physical Model

Learn Parameters of Physical Model

See: O'Gorman & Dwyer (2018), Patel et al. (2015), Tuia et al. (2016), Beucler et al. (2022)

(Guided) Physics-informed feature selection/transformation

Feature selection: Using physical knowledge, causal discovery algorithms…

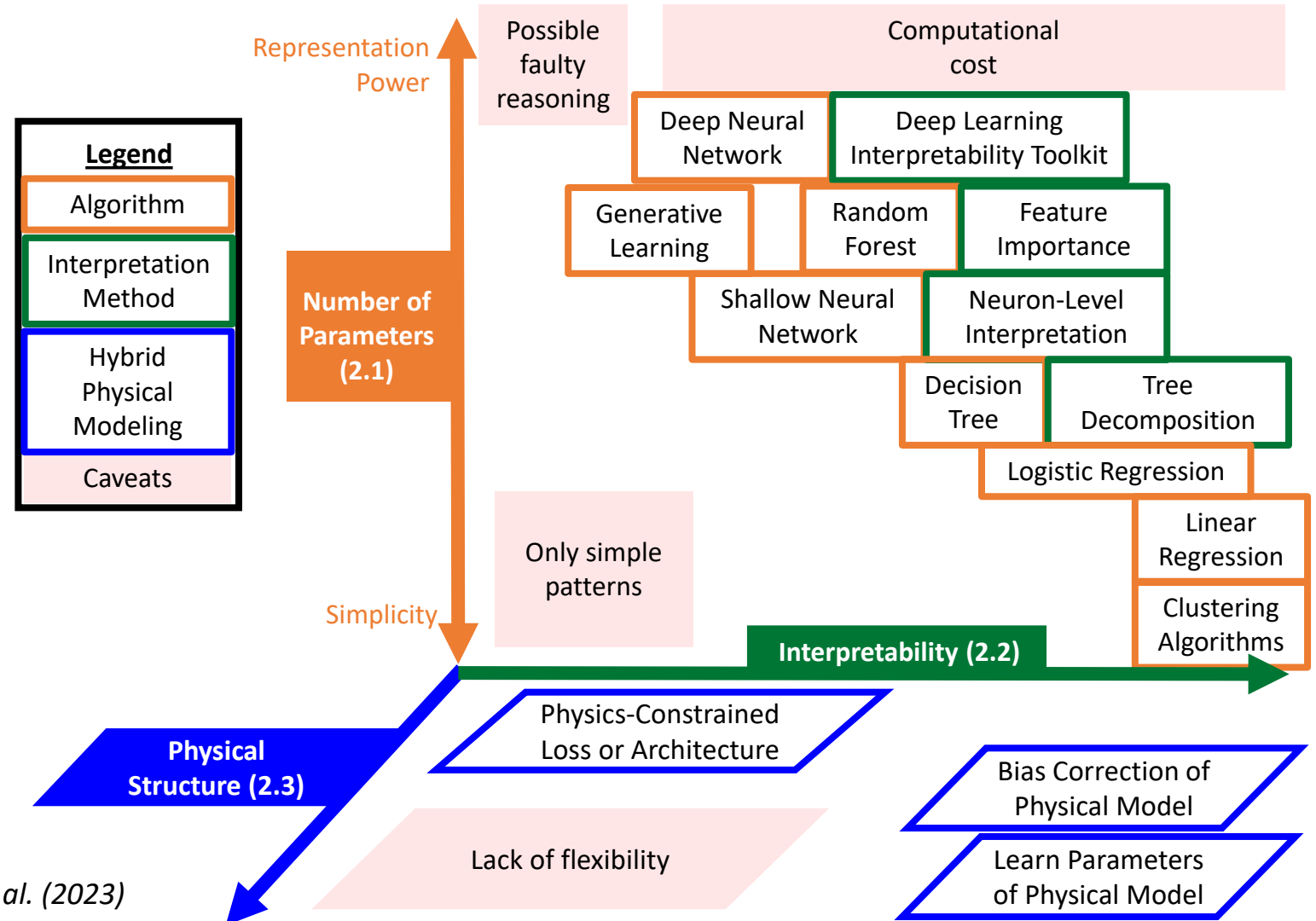


Physics-Guided ML: Add physical structure to restrict ML output to physically-plausible solutions

Physical Structure

Physics-Informed Feature selection /transformation	Physics-Constrained Loss or Architecture	Bias Correction of Physical Model	Learn Parameters of Physical Model

Reviews: Willard et al. (2020), Kashinath et al. (2021), Reichstein et al. (2019), Karpatne et al. (2017), Beucler et al. (2023)



Review:
Beucler et al. (2023)

TROPICANA GitHub Repository:

https://github.com/tbeucler/2024_TROPICANA_ML

TROPICANA GitHub Repository:

https://github.com/tbeucler/2024_TROPICANA_ML