DATAWorks 2025: Interpretable Machine Learning 101

Nikolai D. Lipscomb

Rebecca M. Medlin, Project Leader

DRAFT FINAL

April 2025 | 3004260

Systems and Analyses Center
730 East Glebe Road
Alexandria, VA 22305-3086

IDA

The Institute for Defense Analyses is a nonprofit corporation that operates three federally funded research and development centers. Its mission is to answer the most challenging U.S. security and science policy questions with objective analysis, leveraging extraordinary scientific, technical and analytic expertise.

The Systems and Analyses Center conducted this work under IDA's Central Research Program, C9082, "CRP Statistics WorkGroup." The views, opinions, and findings should not be construed as representing the official position of the U.S. government or the Department of Defense.

Dr. Rebecca M. Medlin, Project Leader
rmedlin@ida.org, (703) 845-6731

Dr. Heather M. Wojton, Director, Operational Evaluation Division
hwojton@ida.org, (703) 845-6811

The IDA Technical Review Committee was chaired by Dr. Heather M. Wojton and consisted of Dr. Dhruv K. Patel, Dr. John T. Haman, Dr. Keyla Pagan-Rivera, and Dr. Rebecca M. Medlin from the Operational Evaluation Division, and Mr. Stuart M. Rodgers from the Science, Systems, and Sustainment Division.

Executive Summary

Traditional machine learning tasks which are not considered generative artificial intelligence are focused on prediction, classification, optimal action selection, and feature extraction problems across various domains such as computer vision, natural language processing, and operations management. Examples of these tasks include automating robotic systems that use computer vision and other spatial sensors to determine the best action to take at the next decision point, using natural language processing algorithms to automatically classify emails as either harmful or benign, and accurately estimating costs of a new, highly complicated venture given information from previous ventures. Computer vision and natural language processing tasks are generally dominated by neural network architectures, particularly transformers, that have reached such high complexity that understanding the full inner-workings of a model is generally considered impossible for a single human. This lack of a comprehensive understanding leads to experimental testing as a means of understanding the model's mechanics; however, a test space in a high-dimensional problem is seldom comprehensive due to an exponentially complicated range of possibilities. This issue frequently leads to neural network models being labelled as "black box models."

In high-stakes decision-making, having wavering confidence in a model's ability to perform correctly and accurately is a dubious proposition. Instead, many practitioners prefer to incorporate *interpretable* models which often provide full understanding of how a model would perform, even on untested data points, due to a comprehensive understanding of the model. A single regression tree, for example, explicitly lists simple criteria that lead to prediction values. Further, interpretable models allow easy communication of model results and structure to a general audience and allow easy diagnosis of problems in both the model and the data, should they arise.

This mini-tutorial, developed for the test and evaluation community attending DATAWorks 2025, focuses on introducing the methods and concepts used in *interpretable machine learning*, particularly for applications that incorporate prediction and classification with tabular data, a field where neural networks regularly underperform compared to more traditional models and ensemble-based methods. This course covers machine learning-wide

concepts such as supervised versus unsupervised learning, online versus offline learning, the curse of dimensionality, bias-variance trade-off, hyperparameter tuning, statistical validation, convexity, and constrained versus unconstrained optimization. Simple interpretable models, such as linear regression for prediction and logistic regression for binary classification, are introduced first, with examples that detail their interpretability. Since complexity (and capability) are generally inversely correlated with interpretability, we briefly discuss how to interpret more complex models, such as ensemble methods. Last, we cover some further reading and relevant research papers for those interested in contributing to future interpretable machine learning research.

This course was originally developed as a mini-tutorial for DATAWorks 2025 attendees; however, there is a plan to offer an expanded version of the course to IDA research staff at a later date which includes technical information and advanced content that was either relegated to the backup slides section or removed from the mini-tutorial. There is current interest within S3D to run some of the topics in short seminars.

The only prerequisites for this course are familiarity with mathematical notation and elementary linear algebra: mainly the fundamentals of matrix/vector operations, matrix inverses, and ill-conditioned matrices.

DATAWorks 2025: Interpretable Machine Learning 101

Nikolai D. Lipscomb

April 2025

Institute for Defense Analyses
730 East Glebe Road ● Alexandria, Virginia 22305

Examples of potential Machine Learning (ML) tasks and data:

Task	Data
Estimate the fuel required for a SAR operation	Vehicle specifications, route details, terrain and weather information
Automatically quarantine malicious emails	Sender information, *free text*†
Choose the best routing action in a logistics network when demands are uncertain	Historical demand information, expected costs/rewards associated with different routing actions
Automatically adjust the trajectory and velocity of an automated vehicle	Images from mounted cameras, local LiDAR maps, other sensor readings, current trajectory and velocity

†: Free text data is unstructured, string-based data that was input freely

LiDAR: Light Detection and Ranging; ML: Machine Learning; SAR: Search and Rescue

Some ML tasks can be complicated due to a variety of reasons

Task	Data	Complication
Estimate the fuel required for a SAR operation	Vehicle specifications, route details, ~~terrain and weather information~~	Insufficient data often leads to poorer estimation capabilities
Automatically quarantine malicious emails	Sender email address, free text	Free text is an abstract data structure—how should we represent it for mathematical decision-making?
Choose the best routing action in a logistics network when demands are uncertain	Historical demand information, expected costs/rewards associated with different routing actions	Historical demand is highly variable, leading to more uncertainty in the quality of a single outcome
Automatically adjust the trajectory and velocity of an automated vehicle	Images from mounted cameras, local LiDAR maps, many other sensor readings, current trajectory and velocity	Spurious signals and contradictions can arise across multiple high-dimensional data sources, particularly with images involved

LiDAR: Light Detection and Ranging; ML: Machine Learning; SAR: Search and Rescue

Complications in ML models lead to follow-up questions and a need to interpret findings/diagnose problems

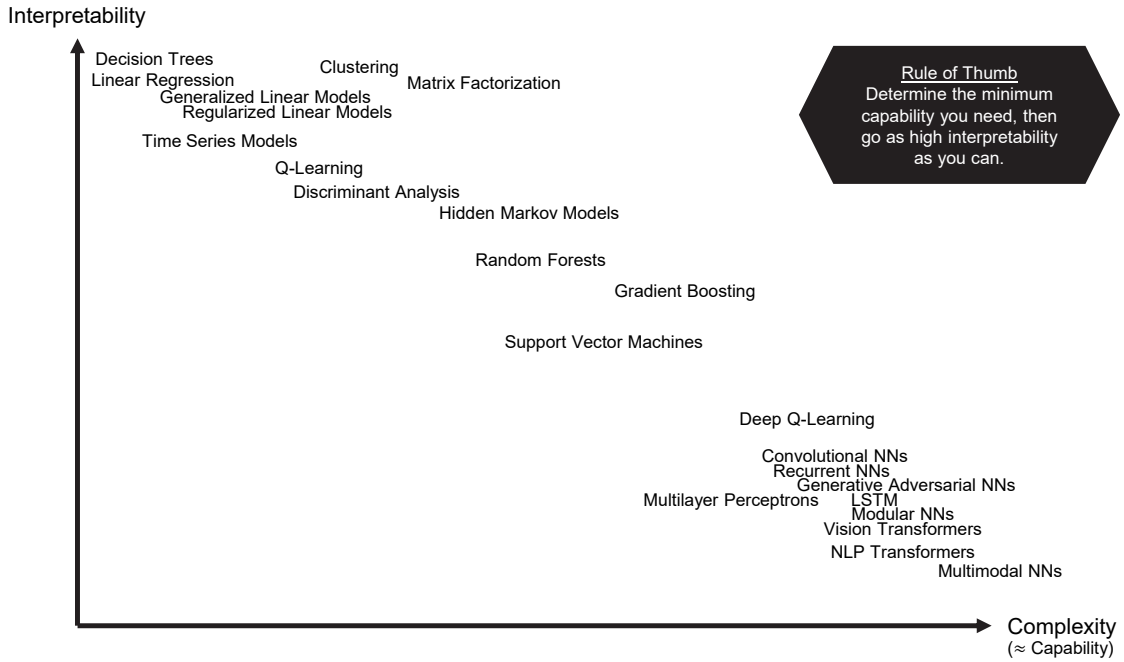
Interpretability in ML is ill-defined; however, a highly-interpretable ML model should have inner mechanics that are well understood by the user. Further, it should be easily explained to an outside audience.

Interpretability and complexity are often competing concepts in ML

Complexity often (but not always) corresponds to enhanced capabilities.

Linear Regression for NLP tasks? Possible, but not likely to be useful.

The chart on the right is subjective, but captures the general complexity/interpretability trade-off.



Rule of Thumb
Determine the minimum capability you need, then go as high interpretability as you can.

LSTM: Long-term Short-term Memory; ML: Machine Learning; NLP: Natural Language Processing; NN: Neural Network

Interpretable models allow for easy communication of findings and quick diagnoses

"Why is the SAR cost estimated to be almost double the last one, despite the length, area, and staffing of the mission being identical?"

"Unlike the last mission, this SAR takes place over a mountainous region, which leads to higher fuel costs. In our estimation model, this more than halves the MPG efficiency of rescue vehicles. Based on historical mountain SAR operations where fuel is the primary driver of cost, we believe this estimate is accurate!"

"This new logistics routing model underperformed relative to the previous year's approach. I think we should roll back to the older modeling approach."

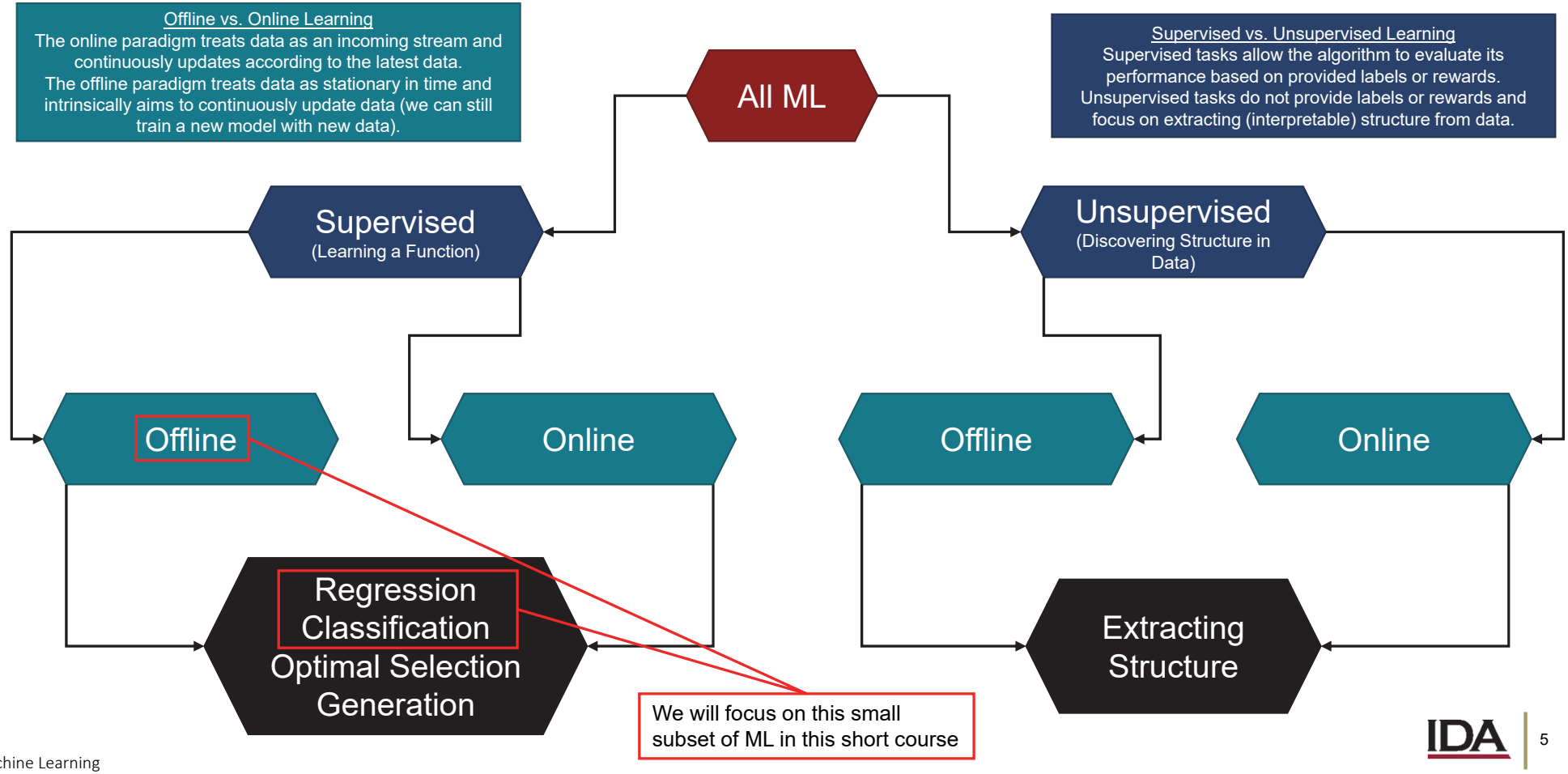
"Our model is trained on several years of historical data; however, this past year was a significant outlier compared to the historical pattern. The older model also would have underperformed in these conditions—even worse than our new method. Further, there is strong evidence that we will return to the usual OPTEMPO. That is, I expect our logistics routing model will dominate the older model."

"My NIKOLIES self-driving car suddenly drove into a pedestrian even though I was sitting at a red light! What happened?"

"The pedestrian was holding up a poster with a green-light traffic signal printed on it. Unfortunately, since the system only uses CV for object detection and recognition, it was tricked by the poster. Pedestrians carrying life-like images of traffic signals had not been explored yet in training. If we incorporated LiDAR, along with additional sensors, we could have recognized the CV signal as false."

CV: Computer Vision; LiDAR: Light Detection and Ranging; MPG: Miles per Gallon; OPTEMPO: Operational Tempo; SAR: Search and Rescue

The scope of ML is massive, we will only look at a small subset of methods in order to understand fundamental concepts in interpretable ML



ML: Machine Learning

We will refer to two running examples in this course. One for regression…

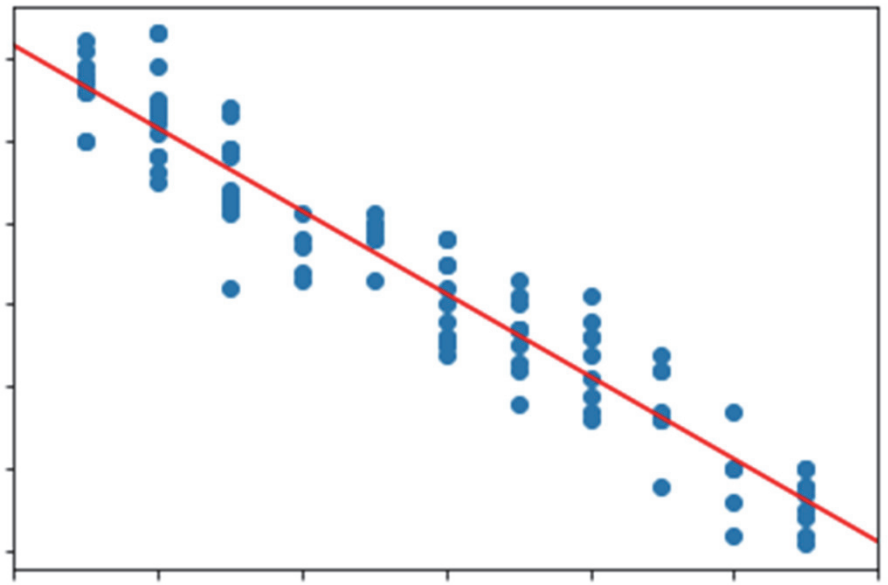
Regression

We wish to estimate the monthly demand for a particular aircraft part.

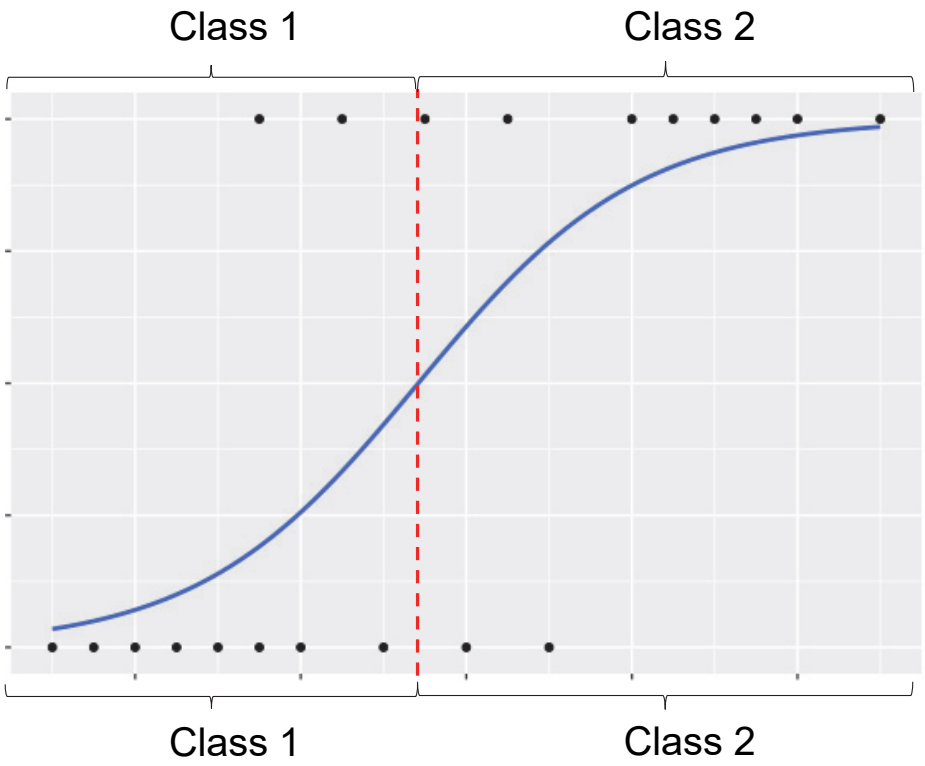
$y_i :=$ the monthly demand for month i.

$X_i :=$ vector of *predictor* variables for month i.
Data such as planned flight hours, number of demands the previous month(s), average temperature that month, etc.

We want a function
$f(X_i) = \hat{y}_i$
Where $\hat{y}_i$ will be good at estimating y_i for *future* months.



...and the other for classification



Classification

We wish to quickly determine if a particular armored platform will result in penetration vs. nonpenetration against standard munitions.

y_i ≔ a penetration or nonpenetration assignment (usually 0 or 1) for platform i.

X_i ≔ vector of *predictor* variables for platform i.
Data such as armor material, thickness, temperature, humidity, munitions velocity, munitions, etc.

We want a function
$f(X_i) = P(y_i = 0) = \hat{p}_i$
Where $\hat{p}_i$ is closer to 1 if $y_i = 0$ and close to 0 if $y_i = 1$.

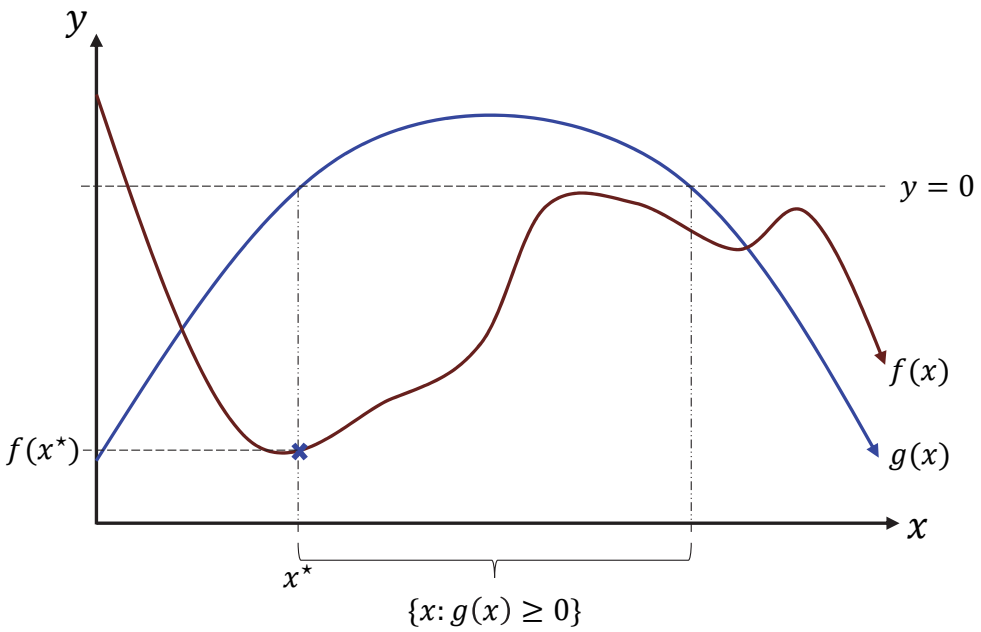
Some important fundamentals of ML are from optimization theory

Optimization

$$\min f(x)$$
$$\text{s.t. } g(x) \geq 0$$

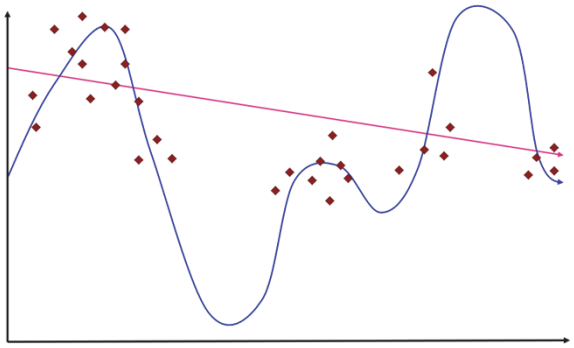
ML models are generally derived from optimization problems that contain a ***loss function*** that we seek to minimize and (possibly) ***constraints*** on what solutions are permissible. If an optimization problem has constraints, then we must restrict our search to the ***feasible set*** of solutions.

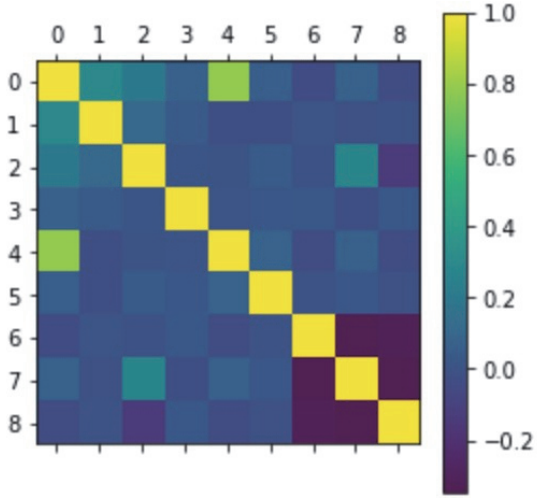
Optimization problems vary in complexity. ***Convexity*** is a geometric concept that relates to both functions and feasible sets and is generally a desired property because it makes finding the best solution to an optimization problem easier, sometimes trivializing the process.



ML: Machine Learning

Statistical fundamentals are also a major aspect of ML





Statistics

Bias-Variance Trade-off: generally we want both consistency and accuracy to be high (low variance and low bias, respectively); however, in a complicated dataspace, one usually comes at the cost of another!

The *Curse of Dimensionality*: when people refer to *big data* or *high-dimensional data*, usually they are referring to the number of variables and not the number of observations in a dataset.† However, it can introduce complications in methods and interpretability as the number of variables in a model grows large. *Dimension reduction* is a key concept in ML and it is intrinsically tied to the bias-variance trade-off.

Model Selection/*Hyperparameter Tuning* and *Testing*/*Validation*: often we compare thousands (or more) of machine learning models. These models can often be determined by changing *hyperparameters* that uniquely vary a common optimization problem. Out of so many ML models, we need to pick the best one. This is performed via proper sampling methods for creating *training* and *validation*/*test* sets to ensure fair comparisons.

†: Usually, observations are plentiful. However, there are unique cases where observations might be limited whereas the number of variables is plentiful. For example, consider MRI scans of individuals. Medical data can be difficult to aggregate due to privacy restrictions and concerns in addition to the costs associated with gathering medical images. However, a scan itself may consist of hundreds of millions of variables (each pixel's numerically encoded value is a variable). This type of data is often handled very delicately with medical imaging SMEs in addition to statistical and ML expertise.

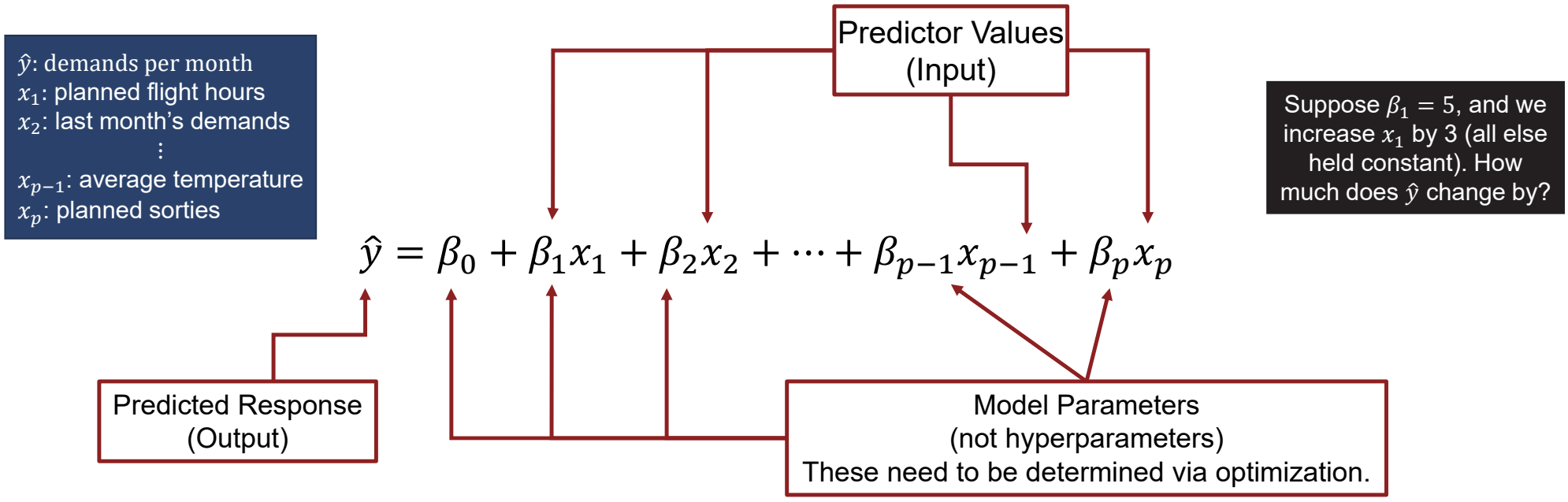
ML: Machine Learning; SME: Subject Matter Expert

Regression

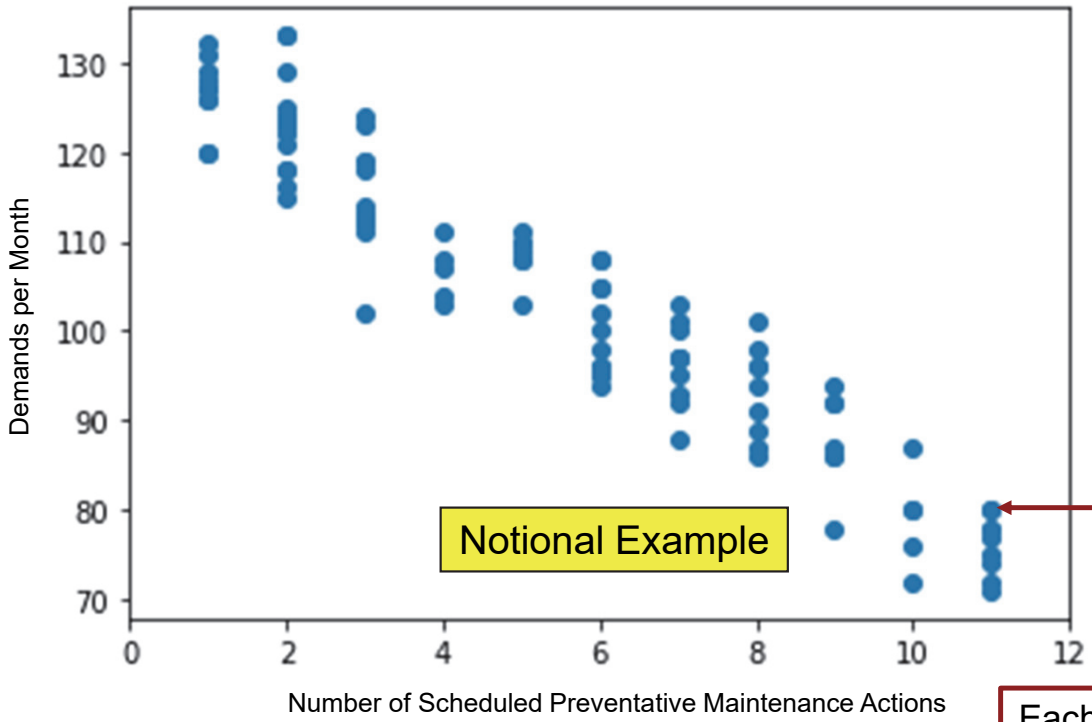
Linear Regression is a good foothold for explaining ML concepts

Predict a numerical response given a set of values for p predictor/feature variables.

Assumes a linear form:



Does a linear relationship seem reasonable?



Response

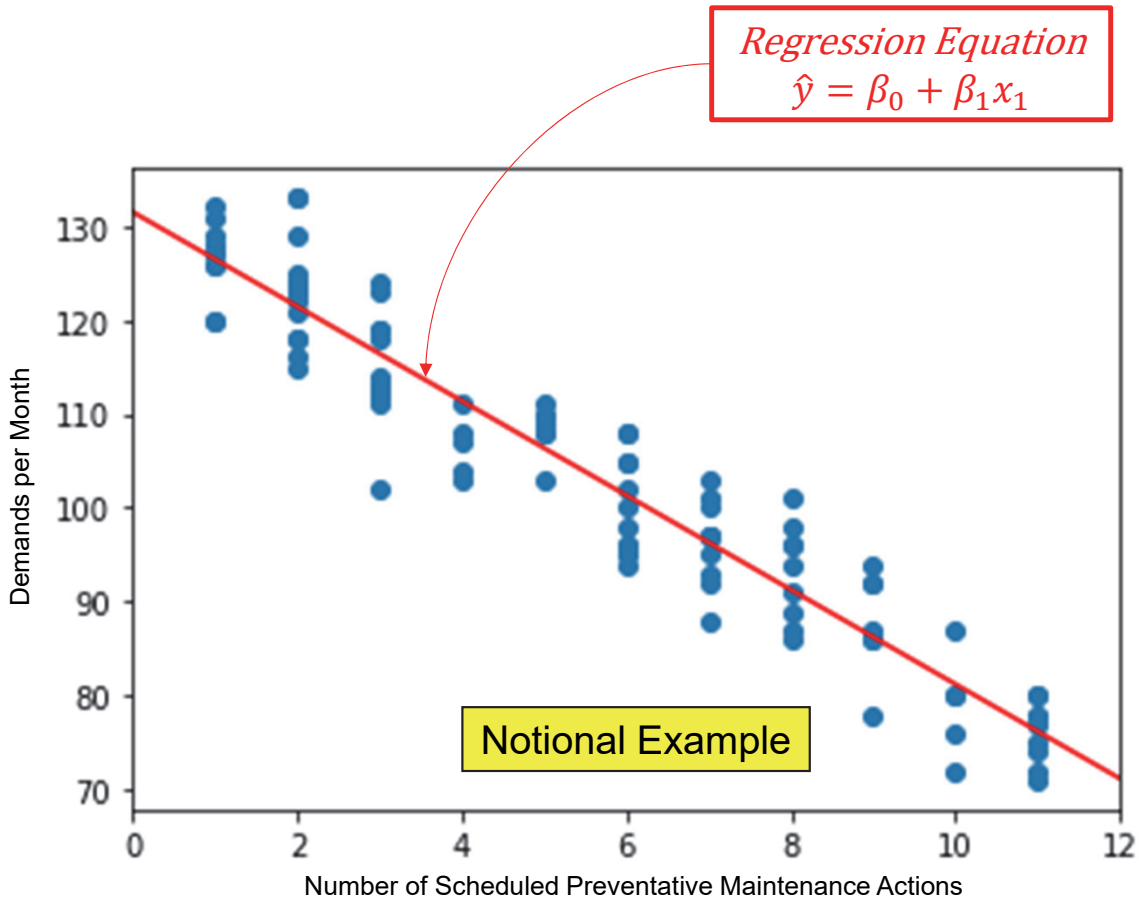
Regression Equation
$\hat{y} = \beta_0 + \beta_1 x_1$

Best choice of line?

Single Predictor

Each dot is an observation:
$(x_{i,1}, y_i)$

In this case, a line of best fit looks good—how does one get the line of best fit?



The line of best fit can be determined by setting up the appropriate optimization problem

$$\min \quad \sum_{i=1}^{n} (\mathrm{y_i} - \hat{y}_i)^2$$

Or equivalently:

$$\min \quad \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_p x_{i,p})\right)^2$$

- y_i : the actual response value of the i-th observation in the training set of size n
- $\hat{y}_i$: the predicted response value of the i-th observation using the linear model
- $x_{i,j}$: the value of the j-th predictor for the i-th observation
- β_j : the j-th model parameter out of $p + 1$ parameters
- *Loss Function*: minimize the sum of squared prediction errors
- No *constraints* or *hyperparameters* in this base model

Rewriting in vector and matrix format:

$$\min \quad \|Y - X\beta\|_2^2$$

- It is a *quadratic program* with no constraints.
- It can be solved ***analytically*** using Vector Calculus and has a unique solution because it is *strictly convex*.
  - Take the derivative with respect to $\vec{\beta}$, set equal to the zero vector and solve…

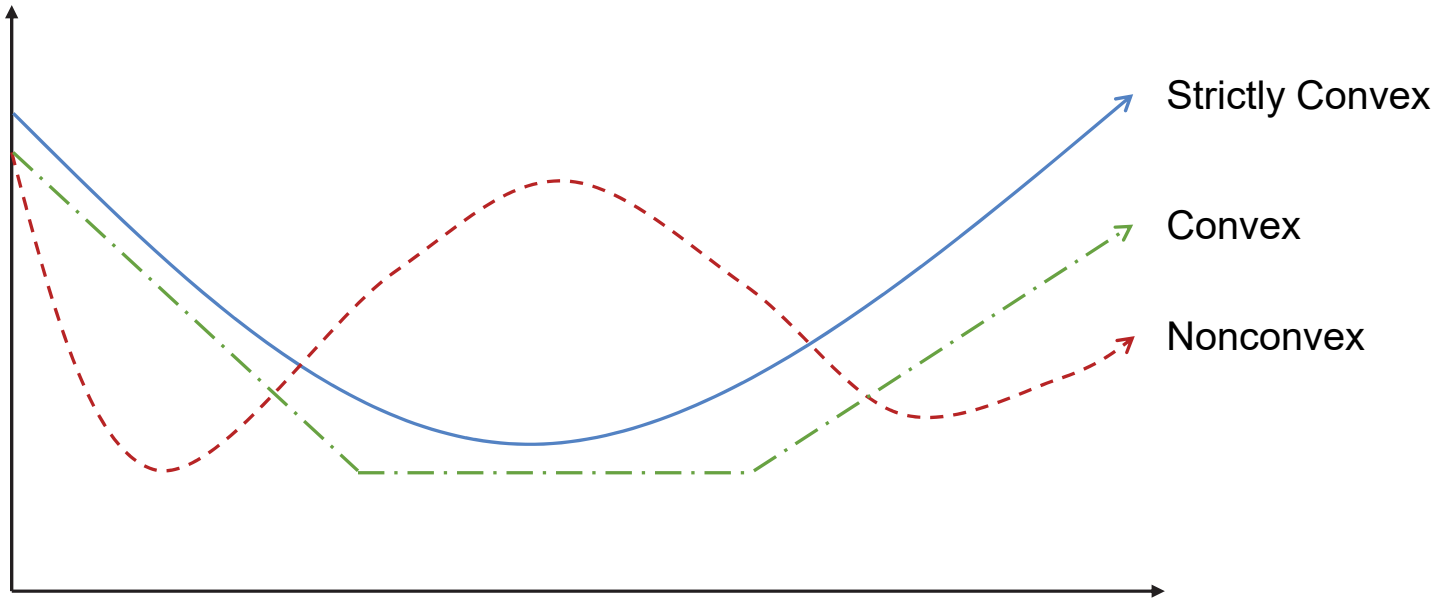
Solution:

Optimal Model Parameter Vector → $\hat{\beta} = (X^T X)^{-1} X^T Y$

Optimal Model Prediction Vector → $\hat{Y} = X(X^T X)^{-1} X^T Y$

We don't want this to be ill-conditioned!
Rarely an issue with few predictors/features.

A quick note on the convexity of functions



What sort of function is easier to minimize?

How do we minimize nonconvex functions?

Evaluating a regression model

Once we have a model, we can examine its performance on *validation/test sets*.

A model should be evaluated on its accuracy, for which there are many *scoring* methods.

These scoring methods generally result from examining the *residuals* (prediction errors).

Remember: we should ultimately be concerned with scores/residuals from validation sets, not training sets!

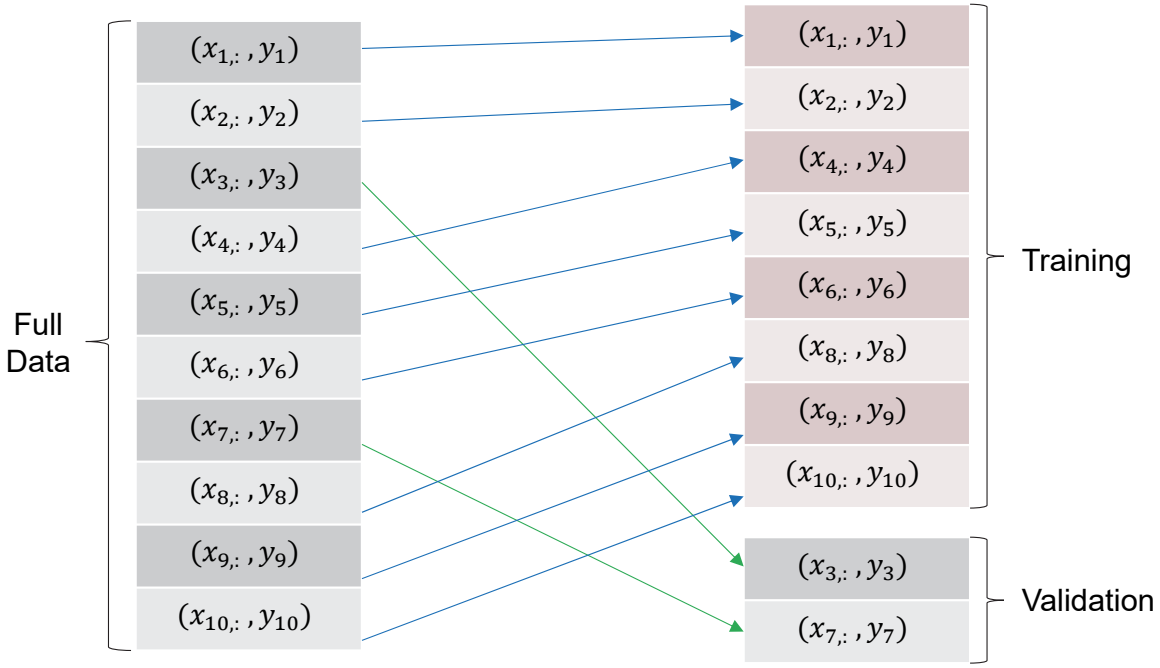
Residuals can be zero, positive, or negative (perfect predict vs. underpredict vs. overpredict).

What we like to see from residuals:

- Unbiasedness (the mean value for residuals is zero).
- Low variance (the spread of the residuals is not too high).

Build test/validation sets using proper sampling methodologies

Any model you build needs to be validated (often multiple times) by sampling test/validation sets from your overall data.



In many regression and classification contexts, it is reasonable to assume that observations are $i.i.d.$, in which case simple random sampling would be appropriate for selecting a validation set.

$i.i.d.$: independent and identically distributed

Scoring is useful for comparing multiple models

Suppose we have 2 predictors:

$$\hat{y} = \beta_0 = \bar{y}$$

(Null Model)

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

(Univariate Model 1)

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

(Univariate Model 2)

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

(Full Model)

One scoring method is to examine the R^2 statistics within test sets

- Most popular metric for "goodness of fit" of a linear regression model:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$$= 1 - \frac{RSS}{TSS}$$

- RSS : Residual Sum of Squares: does this look familiar?
- TSS : Total Sum of Squares: how would you interpret this?

One scoring method is to examine the R^2 statistics within test sets

- R^2:
  - Equal to 1 → perfect prediction: all y values lie exactly on the line of best fit.
  - Equal to $a \in (0,1)$ → $100a\%$ of the variation in the response is captured by the model.
  - Equal to 0 → None of the variation in the response is captured by the model.

- Final Comparison (on validation set):
  - Which model had the highest R^2 score?
  - That's our winner!

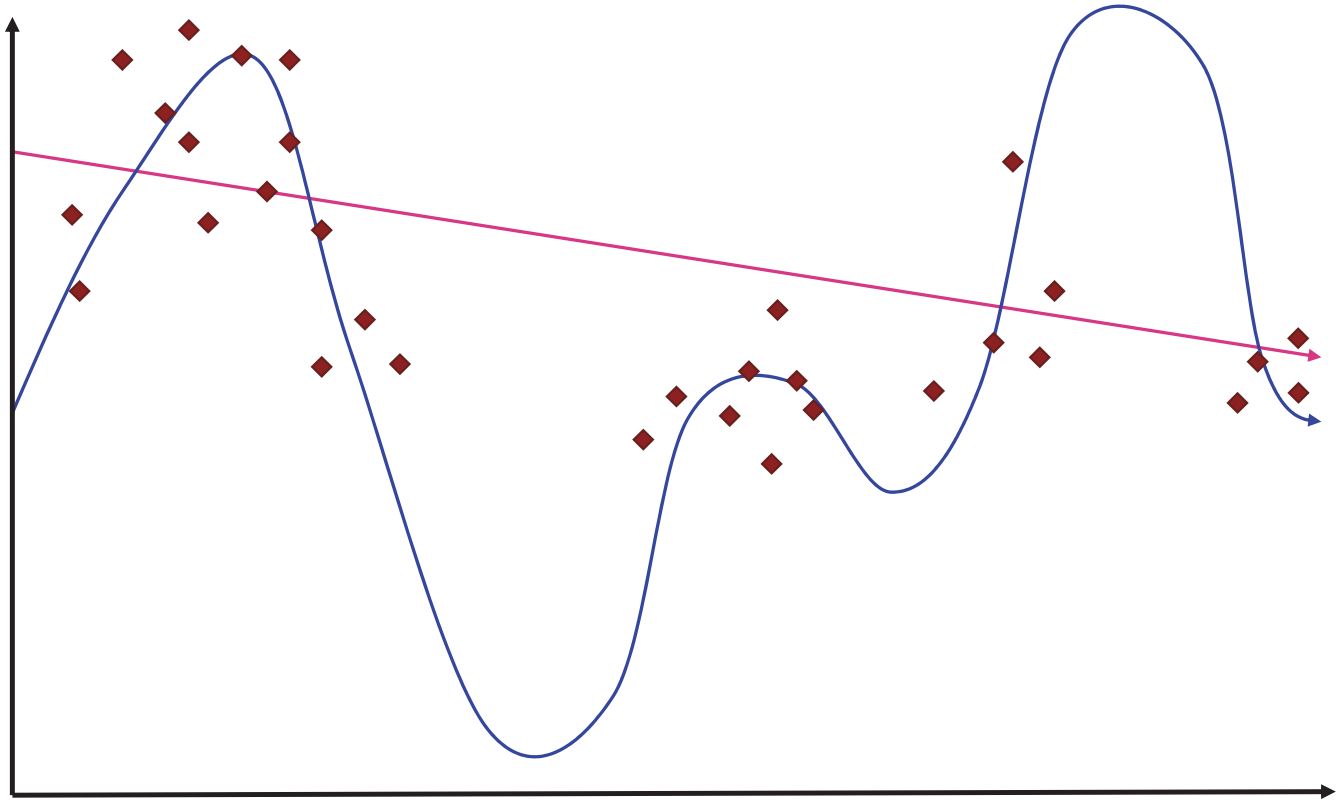
- Within training set:
  - Adding more predictors → higher R^2 score within training set. Always! Why?!
  - Within the training set: $R^2_{submodel} \leq R^2_{fullmodel}$.

When comparing many models, we will often see a trade-off between bias and variance

- One can both overfit and underfit models by incorporating too much complexity (features, in this case).

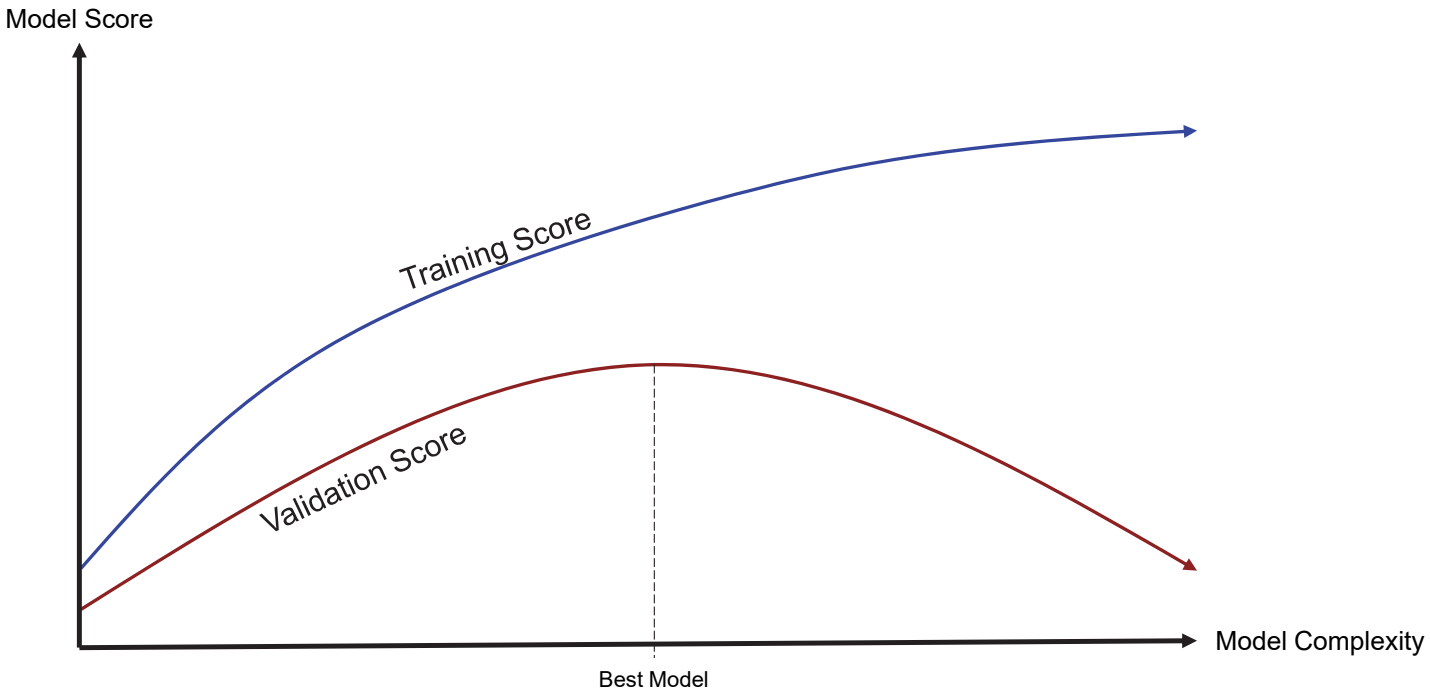
- Think of your goal as building a long-distance rifle:
  - Too many features: High Variance (trajectories within a wide cone, like a shotgun).
    - Better fit to training data.
    - Predictions are highly variable.
    - Captures a lot of the random noise in the training data, misses the true general trend.
  - Too few features: High Bias (misaligned scope).
    - Worse fit to the training data.
    - Predictions have low variability.
    - Misguided predictions by using insufficient information (bias).

Visualization of the bias-variance trade-off



What would you prefer if you had to choose between the two?

As we recall, training R^2 only increases, but we expect validation R^2 to start getting worse beyond a certain complexity threshold



Computational limits are of big consideration in ML; suppose we have 10 predictors

$$\hat{y} = \beta_0$$

(Null Model)

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

(Univariate Model 1)

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

(Univariate Model 2)

$$\vdots$$

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_9 x_9 + \beta_{10} x_{10}$$

(Full Model)

How many total?

ML: Machine Learning

The growth rate is factorial!

$$\sum_{i=0}^{10} \binom{10}{i}$$

$$= \binom{10}{0} + \binom{10}{1} + \binom{10}{2} + \cdots + \binom{10}{9} + \binom{10}{10}$$

$$= 1024$$

Variable selection has many solution approaches

- Comparing all 1024 models will be cumbersome, albeit not completely terrible.
  - 10 predictors is not a lot, what about hundreds, or thousands?
  - Factorial growth!
  - 100 predictors gives us approximately 1.27×10^{30} models!

- Variable Selection Methodologies:
  - Step-wise Selection:
    - Statistics-based: Depends on distributional assumptions.
    - Cross-validation-based: No assumptions, just based on estimated predictive performance.
  - Regularization:
    - Alteration of the optimization problem (loss or constraints).
    - Introduces hyperparameters, requires tuning via cross-validation.
  - Matrix Factorization:
    - Advanced topic: requires lots of linear algebra knowledge.
    - Many factorization approaches fall within *unsupervised learning* approaches.
    - Reduces dimensionality by redefining the basis of your data (change of basis + taking a subset of new basis vectors).
  - Many more!

Computational improvement from forward selection

(ex) 10 predictors:

1. Exhaustive comparison:
   1024 fits to training data.
2. Forward Selection w/ 5-fold cross-validation:
   At most: $(10 + 9 + \cdots + 2 + 1) \times 5 + 1$ = 276 fits to training/subtraining data.

General:

1. Exhaustive comparison with p predictors:

$$\sum_{i=0}^{p} \binom{p}{i} = \sum_{i=0}^{p} \frac{p!}{i!(p-i)!}$$

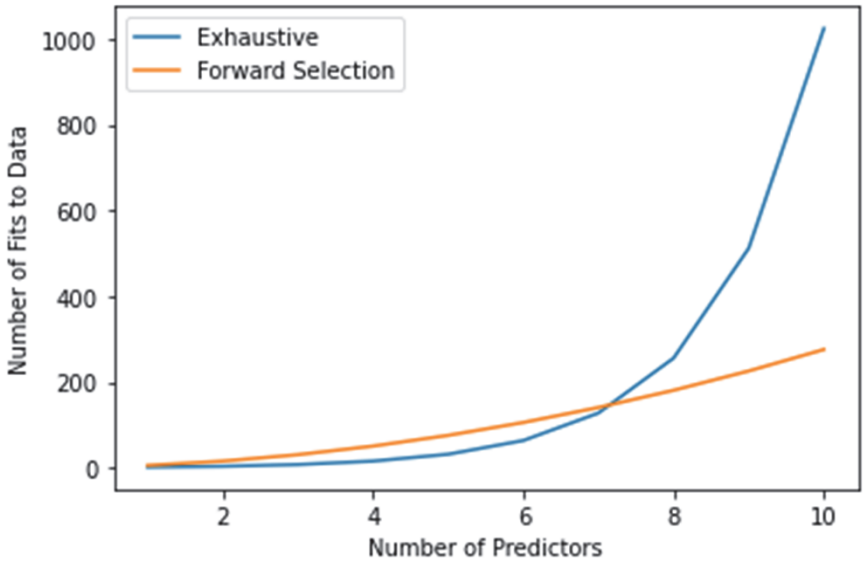
Factorial Growth!

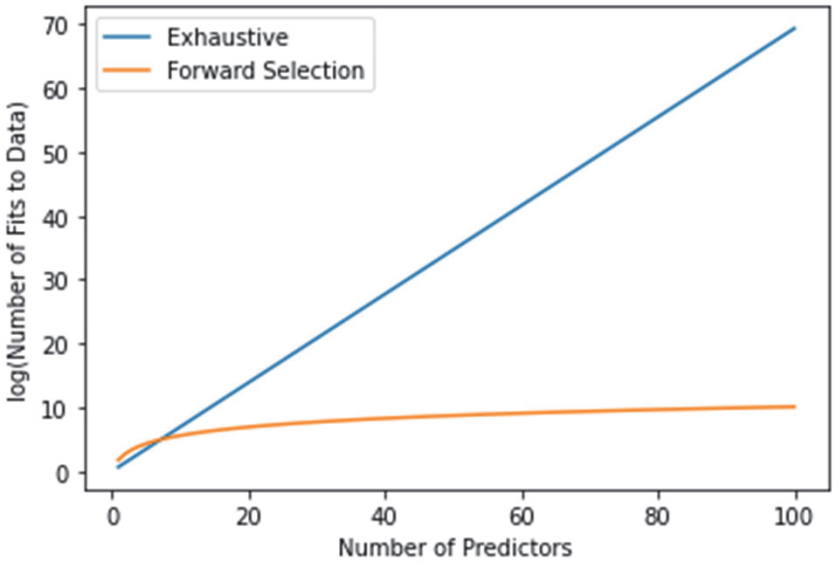
1. Forward Selection w/ p and K-fold cross-validation:

At most: $1 + \mathrm{K} \sum_{i=1}^{p} i = 1 + K\left(\frac{p(p+1)}{2}\right)$

Quadratic Growth!

Log-scale truly reveals the massive gap in computation between the two approaches for larger predictor counts





Note: there is no guarantee a step-wise approach, such as forward selection, will outperform an exhaustive search, since it is a greedy algorithm. However, in practice, is still performs very well.

Regularization Approaches in Regression

Linear correlation between a response and predictor is desired, but between predictors generally leads to an increase in variance

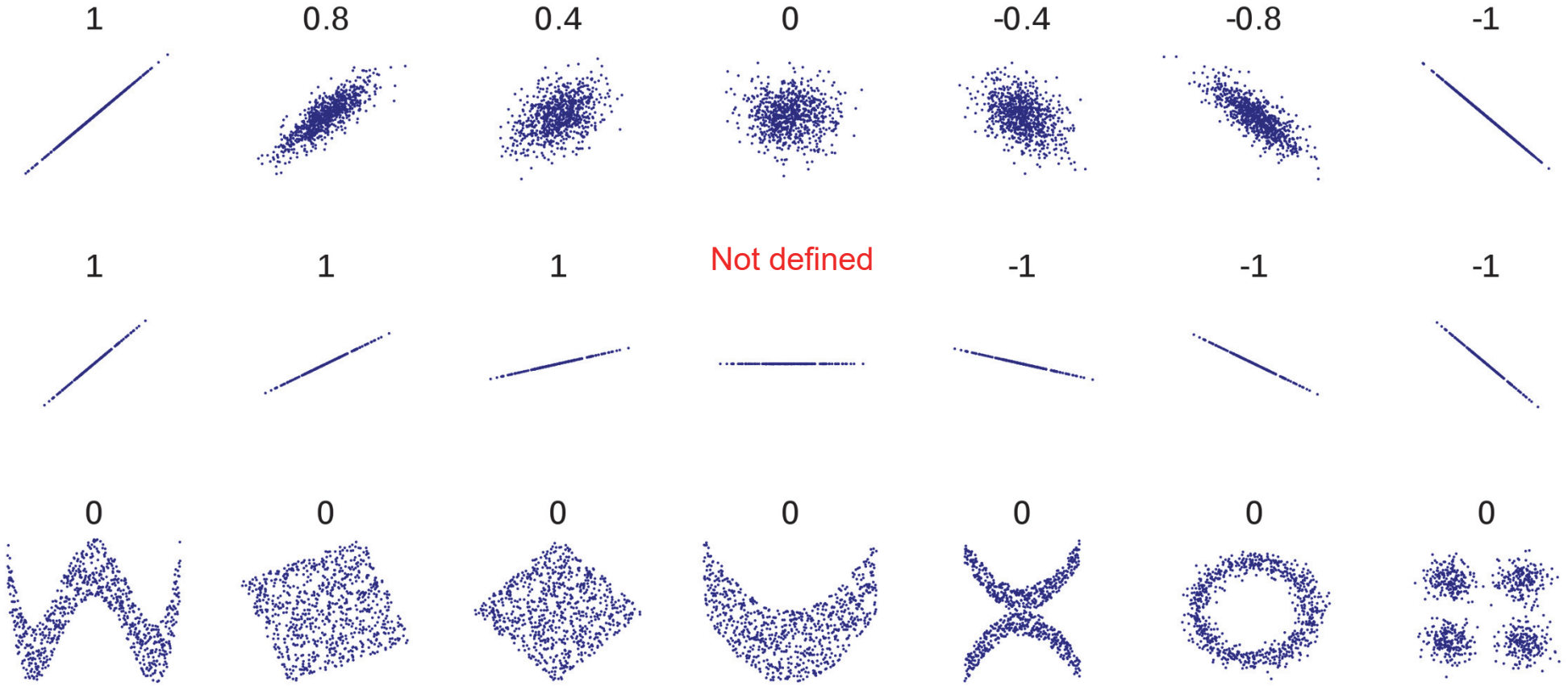
- Given a numerical set of predictors, we can look at the correlation between them
- This can help with detecting multicollinearity: a problem that occurs when predictors are highly correlated (recall our discussion of matrix-inversion and numerical issues)
  - 2+ predictors perfectly correlated: $X^T X$ is singular
  - 2+ predictors have high correlation: $X^T X$ may be ill-conditioned

- Pearson Correlation:

$$r_{i,j} = \frac{\sum_{k=1}^{n}(x_{k,i} - \bar{x_i})(x_{k,j} - \bar{x_j})}{\sqrt{\left[\sum_{k=1}^{n}\left(x_{k,i} - \bar{x_i}\right)^2\right]\left[\sum_{k=1}^{n}\left(x_{k,j} - \bar{x_j}\right)^2\right]}}$$

- Fun fact: squaring this gives us R^2

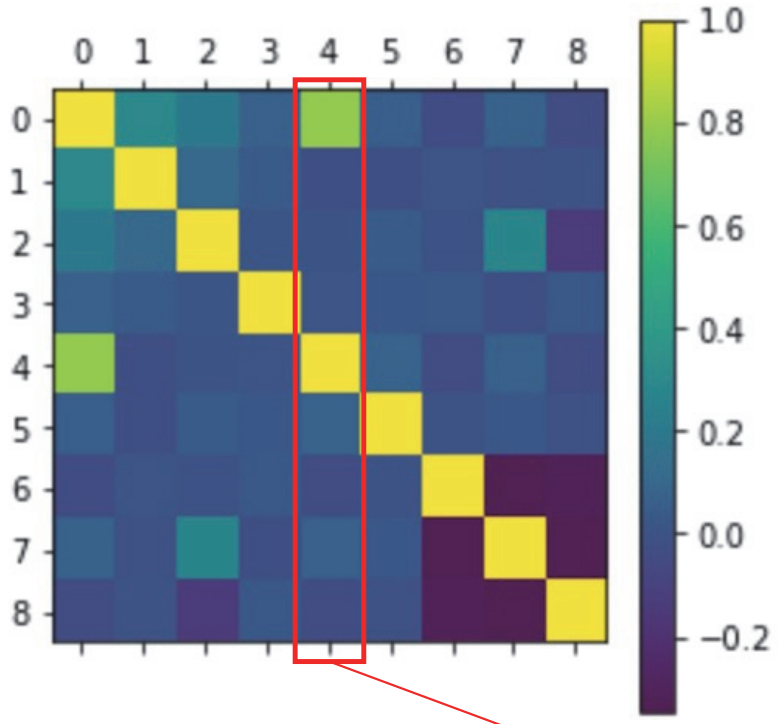
Visualizing correlation



Correlation matrices can help identify problem variables at certain scales

For a set of p-features/predictors, the correlation matrix can be represented as:

Notional Example



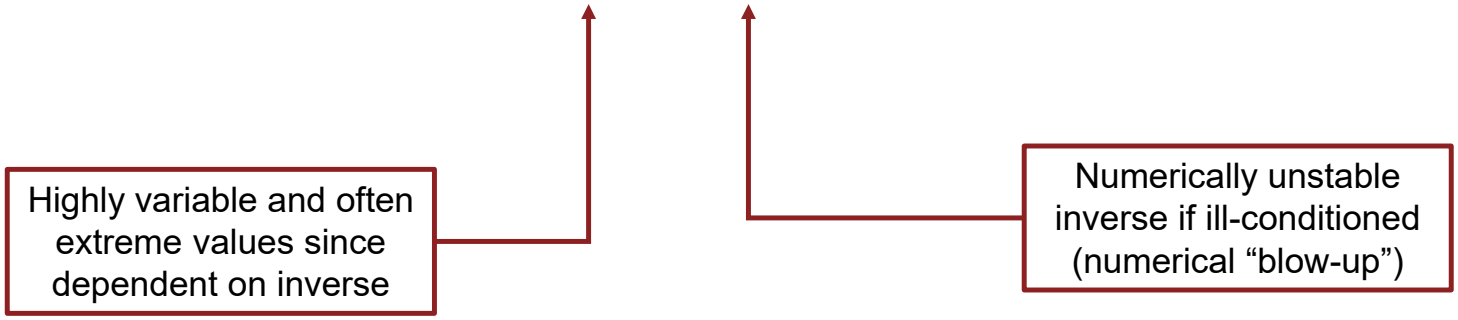
Main diagonal is all ones. Why?

Let's examine this column, what row would we associate it with?

Regularization is an alternative approach to variable selection methodologies that are useful when the dimensionality is large

- As data dimensionality increases, the risk of overfitting and ill-conditioning increases.
  - Big data is in the thousands/hundreds-of-thousands/millions of predictors!
- Regularization is an approach to bound the values of model parameters.
  - Reduces variance of model predictions (bias-variance trade-off) but introduces bias by constraining variables.
- Some regularization methods give us variable selection as a caveat!
- Recall:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$



Regularization can be performed via constraining the feasible set or be a penalty function

- Unconstrained:

$$\min \ \|Y - X\beta\|_2^2 + \lambda f(\beta)$$

Nonnegative Hyperparameter

Penalty Function

$f(\cdot)$ is some measure of the magnitude of $\vec{\beta}$.

- Constrained:

$$\min \ \|\vec{Y} - X\beta\|_2^2$$

$$subject\ to \quad f(\beta) \leq M$$

Positive Hyperparameter

Constraint

Regarding Hyperparameters vs. Model Parameters:

- Model Parameters (β) are derived by the model and establish a relationship between the response and predictors/features.
  - Generally a functional relationship: $f(X|\beta) = \hat{y}$ ← predicted value for response.
  - β is the solution to the optimization problem.

- Hyperparameters (λ) are *fixed* parameters of the optimization problem itself.
  - They modify the optimization problem (either loss function or constraints) and thus lead to different optimal values for β.
  - We place hyperparameters based on certain goals.
  - Hyperparameter tuning takes place outside the loss-function optimization problem—in a sense, it is an "outer" optimization, with the loss-function optimization problem being an "inner" optimization.

A general rule of thumb is that unconstrained optimization is easier than constrained optimization (no need to worry about the feasible region)

- We will focus on loss functions of this form:

$$\min \ \|Y - X\beta\|_2^2 + \lambda f(\beta)$$

L2 Regularization (Ridge Regression)
$f(\beta) = \|\beta\|_2^2$

$$\|\beta\|_2^2 = \sum_{i=0}^{p} \beta_i^2$$

L1 Regularization (LASSO)
$f(\beta) = \|\beta\|_1$

$$\|\beta\|_1 = \sum_{i=0}^{p} |\beta_i|$$

Standardization is important for regularization methods!

$$\min \ \|Y - X\beta\|_2^2 + \lambda f(\beta)$$

When we perform regularization, we must standardize all the variables, including the predictor.

Why?

For most regularization, we are pulling the model parameters to 0 as λ gets larger.

(ex) $\vec{Y}$ is super massive. How does failing to standardize affect predictions?

(ex) X_i is super tiny. How does failing to standardize affect predictions?

A sum of convex functions is also convex, so Ridge Regression is a convex minimization problem

$\|Y - X\beta\|_2^2 + \lambda\|\beta\|_2^2$ is convex!

In fact, since it is in a nice, differentiable quadratic form, the global minimum can be found via Vector Calculus!

$$\hat{\beta}_{ridge}^{\lambda} = \left(X^T X + \lambda I_p\right)^{-1} X^T Y$$

$$\hat{Y}_{ridge}^{\lambda} = X\hat{\beta}_{ridge}^{\lambda} = X\left(X^T X + \lambda I_p\right)^{-1} X^T Y$$

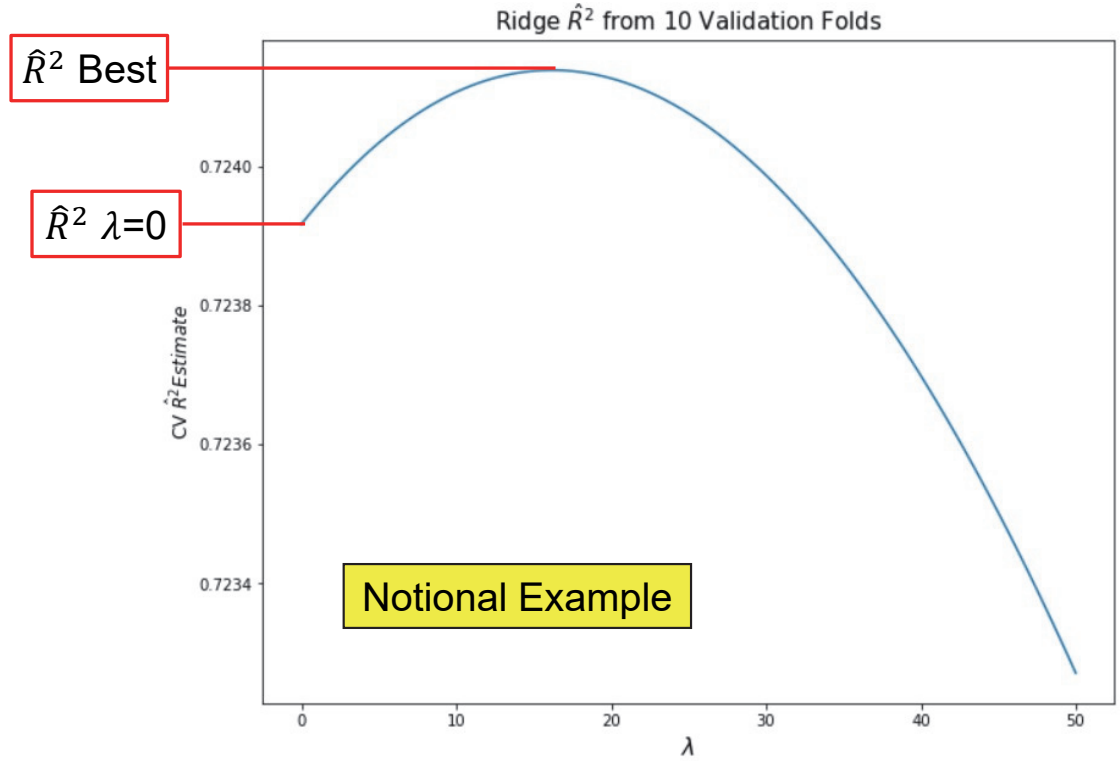
Tuning λ is done by testing it across many *validation* sets

λ should not just be arbitrarily chosen. It should be chosen via cross-validation!

Create a list/array of λ values to try.

For each λ, estimate $\hat{R}^2$ from cross-validation.

Pick the λ with the largest $\hat{R}^2$ value.



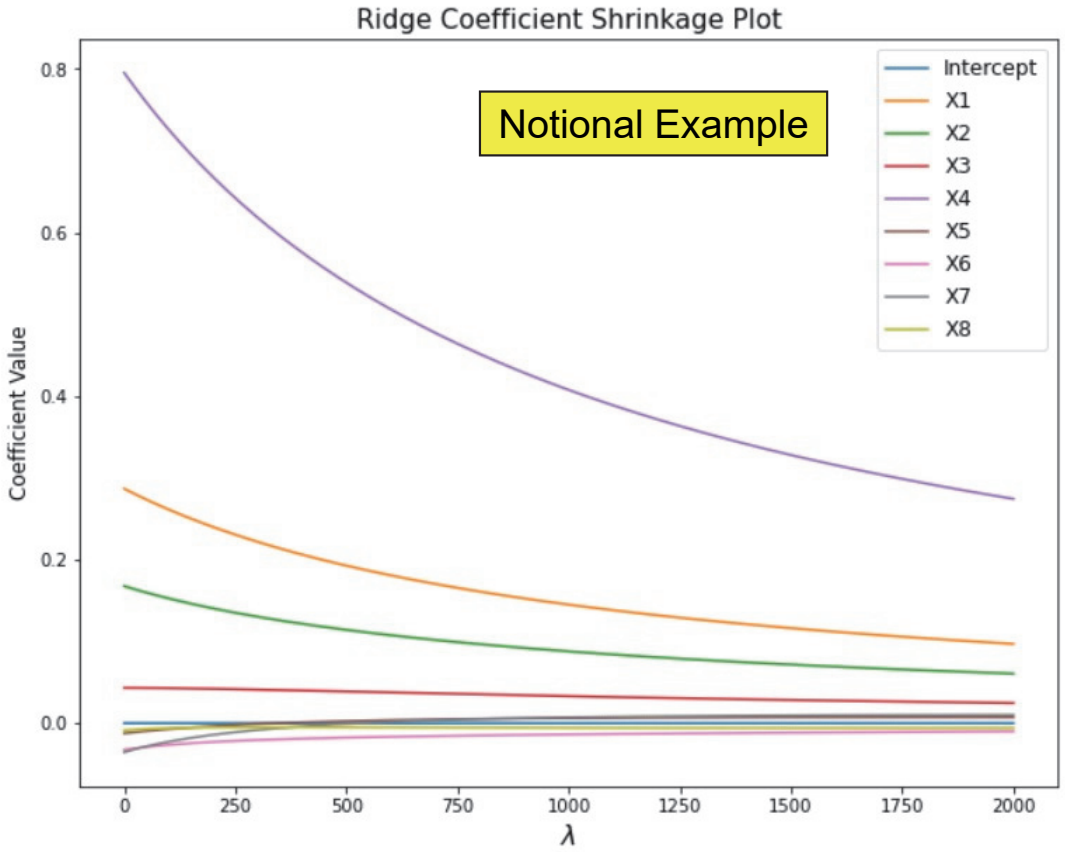
Cross-validation is used to test different values of the hyperparameter while capturing distributional robustness across the training data

Training sorted into 4 random folds.

Note: Folds are generally of equal size, when possible. Okay if only approximate equality (a prime number of observations, for example).

Ridge regression is known as a *shrinkage* method; parameters converge to zero as $\lambda \to \infty$

$$\lim_{\lambda \to \infty} \left\| \vec{\beta} \right\|_2^2 = 0.$$



This is a plot of the values of β_i for $i = 0, \ldots, p$. Not of the values of X_i which are random.

Do not forget that the variables are all standardized, which affects model interpretation!

The Least Absolute Shrinkage and Selection Operator (LASSO), while convex, doesn't have a nice closed-form solution

$$\min \ \|Y - X\beta\|_2^2 + \lambda\|\beta\|_1$$

$\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1$ is also a convex function.

However, in general, due to the non-differentiability of the absolute value function, we use numerical methods to solve.

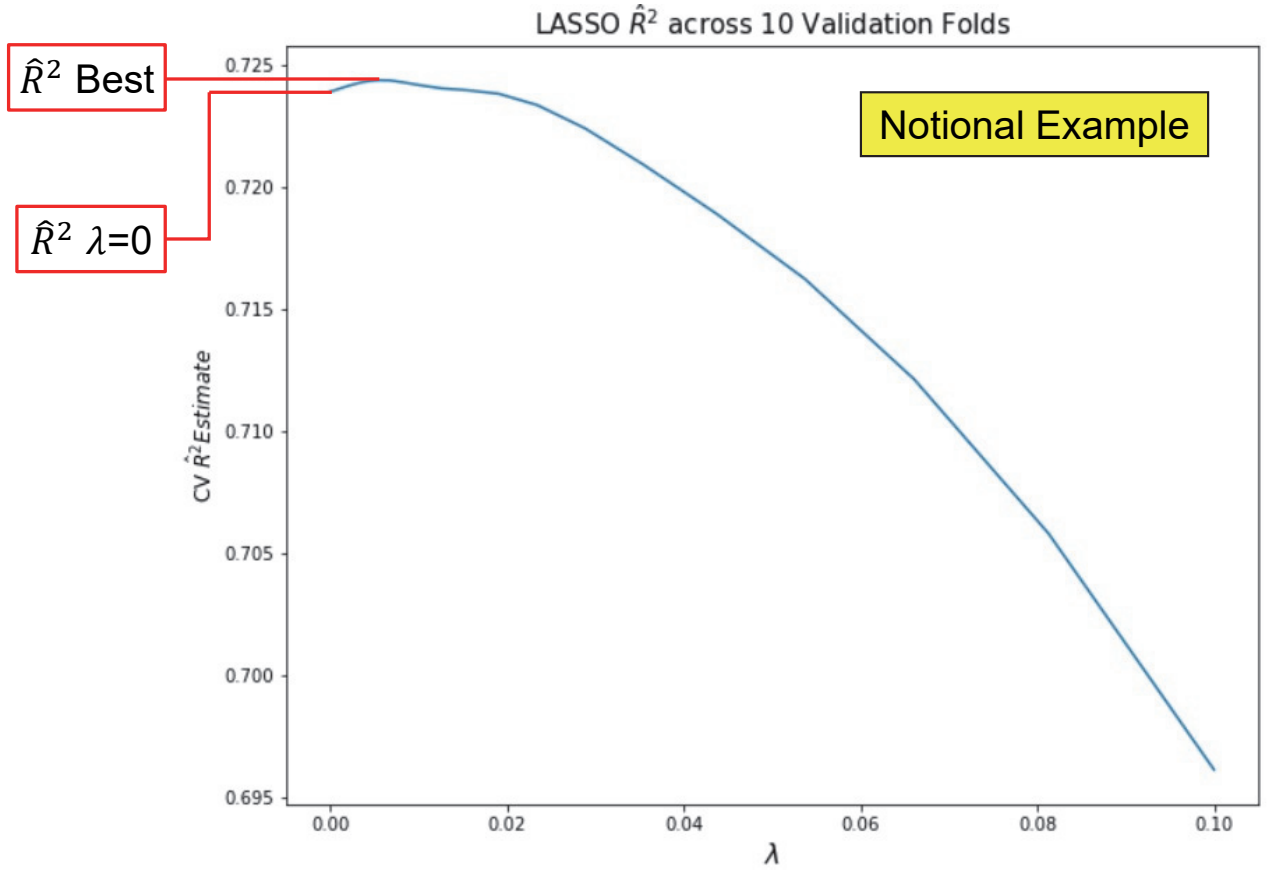
$\hat{Y}_{LASSO}^{\lambda} = X\hat{\beta}_{LASSO}^{\lambda}$ ← No general analytical expression.

Tuning the hyperparameter is business as usual with cross-validation

Create a list/array of λ values to try.

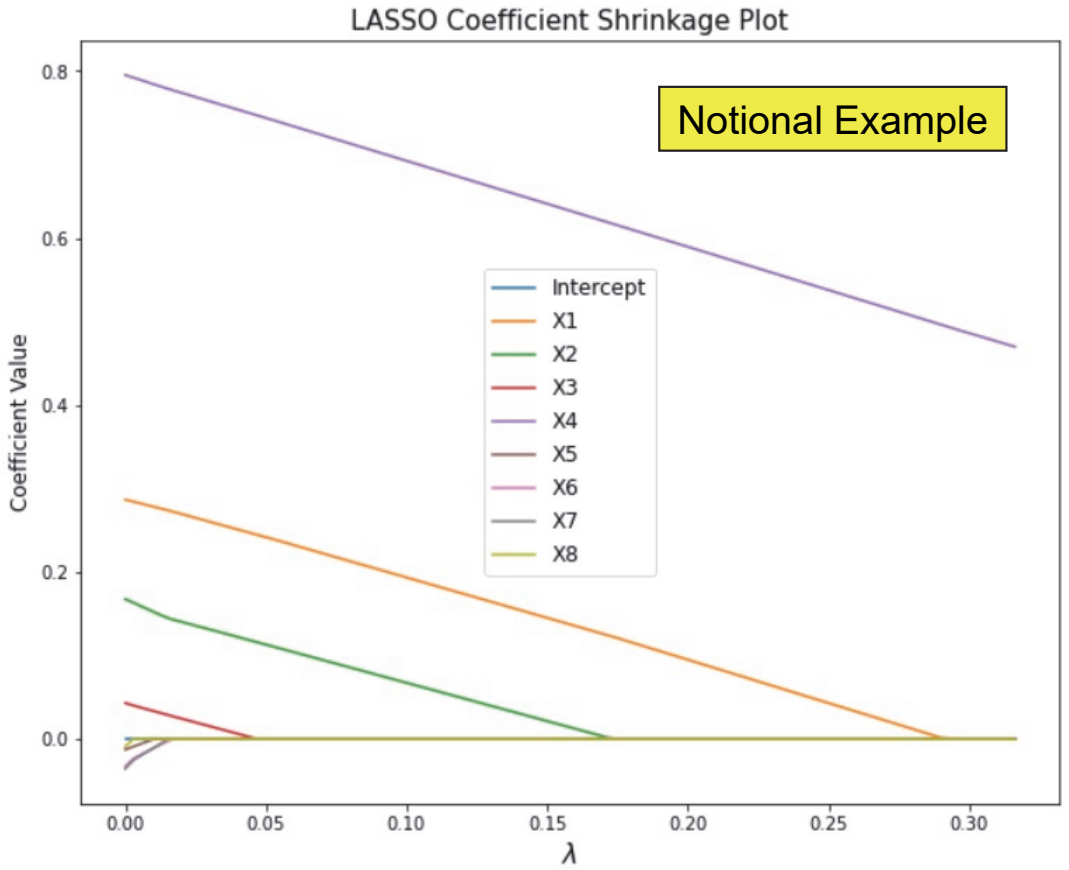
For each λ, estimate $\hat{R}^2$ from cross-validation.

Pick the λ with the largest $\hat{R}^2$ value.



LASSO: Least Absolute Shrinkage and Selection Operator

LASSO is considered a selection operator since coefficients are shrunk to zero in order of least importance

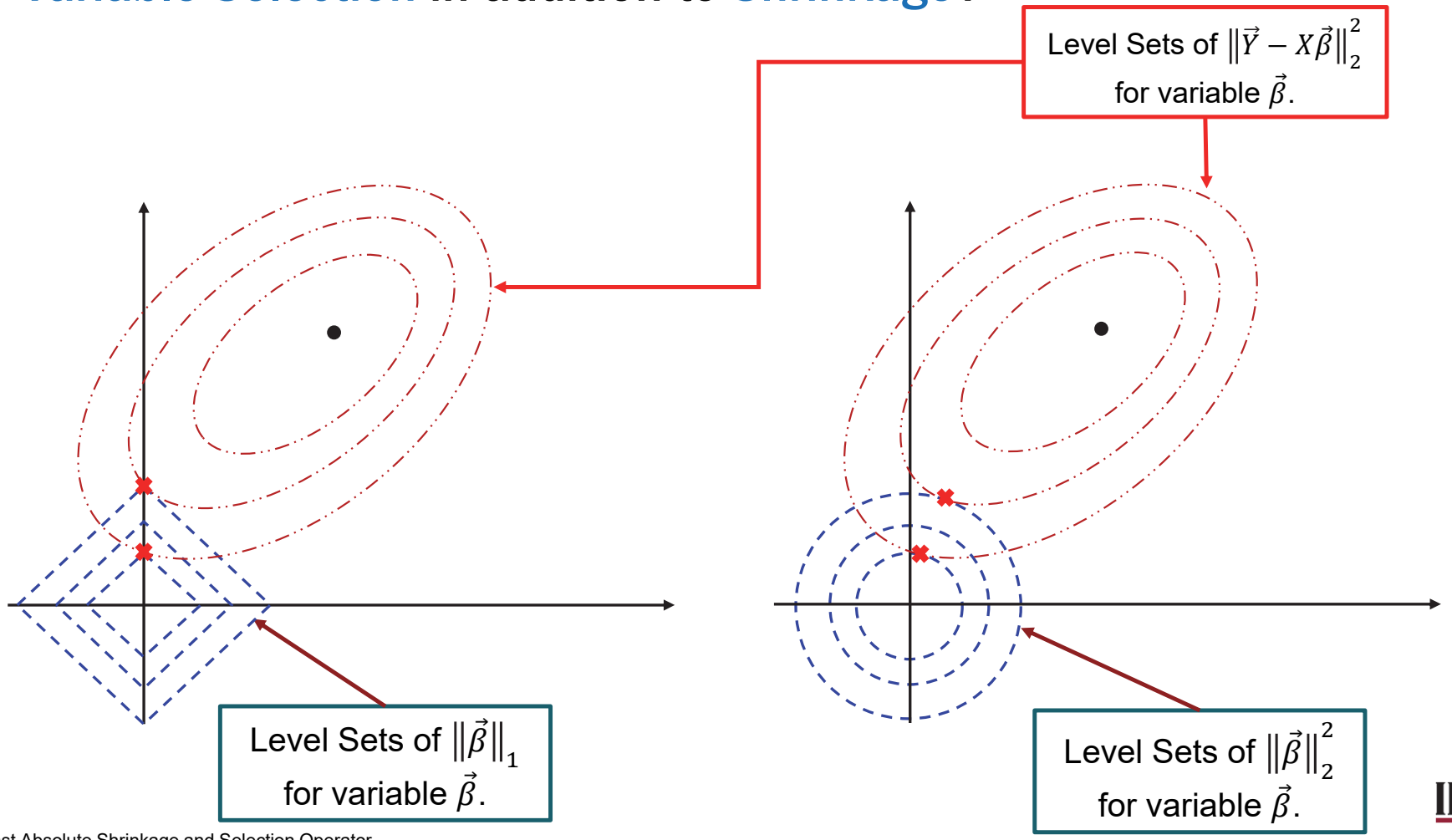


This is a plot of the values of β_i for $i = 0, \ldots, p$. Not of the values of X_i which are random.

Do not forget that the variables are all standardized, which affects model interpretation!

LASSO: Least Absolute Shrinkage and Selection Operator

Why does switching from Ridge Regression→ LASSO give us Variable Selection in addition to Shrinkage?



LASSO: Least Absolute Shrinkage and Selection Operator

Classification

We will recall our binary classification problem as motivation

Classification

We wish to quickly determine if a particular armored platform will result in penetration vs. nonpenetration against standard munitions.

y_i ≔ a penetration or nonpenetration assignment (usually 0 or 1) for platform i.

X_i ≔ vector of *predictor* variables for platform i.
Data such as armor material, thickness, temperature, humidity, munitions velocity, munitions, etc.

We want a function
$f(X_i) = P(y_i = 0) = \hat{p}_i$
Where $\hat{p}_i$ is closer to 1 if $y_i = 0$ and close to 0 if $y_i = 1$.

Logistic regression, a type of generalized linear model, incorporates a familiar form: a linear combination of predictors weighted by parameters

Correctly classify a binary response variable given a set of values for p predictor/feature variables.

Assumes a logistic form with a linear model in the exponent:

$$\hat{p} = \frac{1}{1 + \exp\left(-\left(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{p-1} x_{p-1} + \beta_p x_p\right)\right)}$$

Predicted Response (Output)
$\hat{p} \in (0,1)$

$\hat{p}$: penetration probability
x_1: armor density
x_2: armor thickness
⋮
x_{p-1}: surface temperature
x_p: local humidity

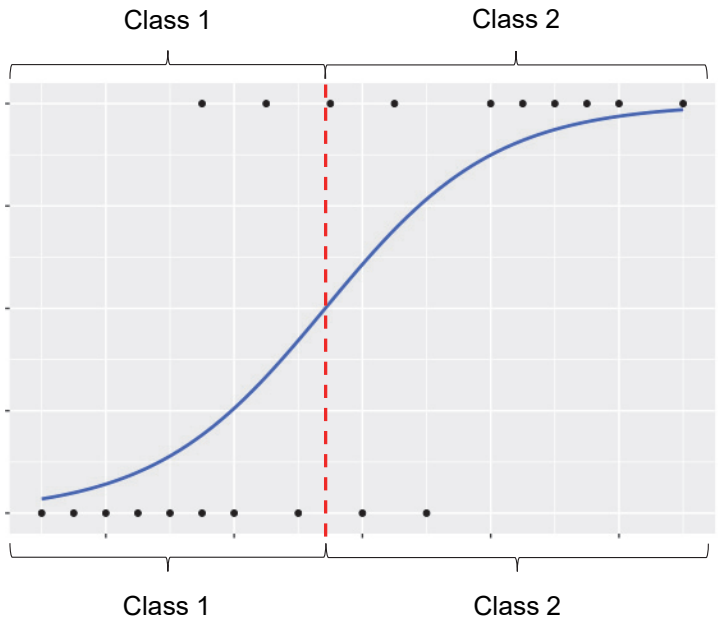
Suppose $\beta_1 = -2$, and we increase x_1 (all else held constant). Does $\hat{p}$ increase or decrease?

Going from a probabilistic output to a classification

$y_i \in \{0,1\}$ ← Binary Categorical Response for observation i in training data.

$\hat{p}_i \in (0,1)$ ← Logistic fitted value for observation i in training data.

Need a decision rule for an observation i with features $x_{i,1}, \ldots x_{i,p}$!



$$P(Y = 1) = \hat{p}_i = \frac{1}{1 + \exp\left(-\left(\beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p}\right)\right)}$$

Popular Decision Rule

$\hat{p}_i < 0.5$ → $\hat{y}_i = 0$

$\hat{p}_i \geq 0.5$ → $\hat{y}_i = 1$

Image Credit: Wikipedia; Author: Canley; <https://commons.wikimedia.org/wiki/File:Exam_pass_logistic_curve.svg>; altered: axis labels and title cropped, class separation and labels added; Licensing: <https://creativecommons.org/licenses/by-sa/4.0/>

We need to be careful with our loss function to ensure convexity when we are able to†

What is a good loss function?

One Idea:

$$\min \sum_{i=1}^{n} (y_i - \hat{p}_i)^2$$

Seem reasonable?

†: Some machine learning models, such as many deep learning architectures, do not have a convex loss function, in which case there may be no guarantee on global optimality. Still, many local optima can be "good enough" for various tasks.

The squared-error loss function will be nonconvex in this scenario

Convex Alternative (minimized via gradient descent-type algorithm):

$$\min -\sum_{i=1}^{n}[y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)]$$

We use the convention $\log(x) = \log_e(x)$.

Remember, the minimization is with respect to β and

$$\hat{p}_i = \frac{1}{1 + \exp\left[-\left(\beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p}\right)\right]}$$

Does this make sense as a classification loss function?

Logistic regression has nice interpretability through the generalized linear model format

Segment the model as follows:

$$\hat{p}(\mu) = \frac{1}{1 + e^{-\mu}}$$

where

$$\mu = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

is a linear model.

Consider:

$\hat{p}(\mu)$ is an invertible function.

Find $\hat{p}^{-1}(\mu) \rightarrow \mu = f(\hat{p})$

Solution

$$\mu = \log\left(\frac{p}{1-p}\right)$$

"log-odds"

Interpreting odds. What are "1:1 odds", "3:1 odds", "1:2 odds"?

So:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

x_1: armor density
x_2: armor thickness
⋮
x_{p-1}: surface temperature
x_p: local humidity

Base log-odds

Marginal change in log-odds with respect to x_1.

Marginal change in log-odds with respect to x_p.

Note: you can impose regularization on logistic regression too! We only standardize predictors in this case.

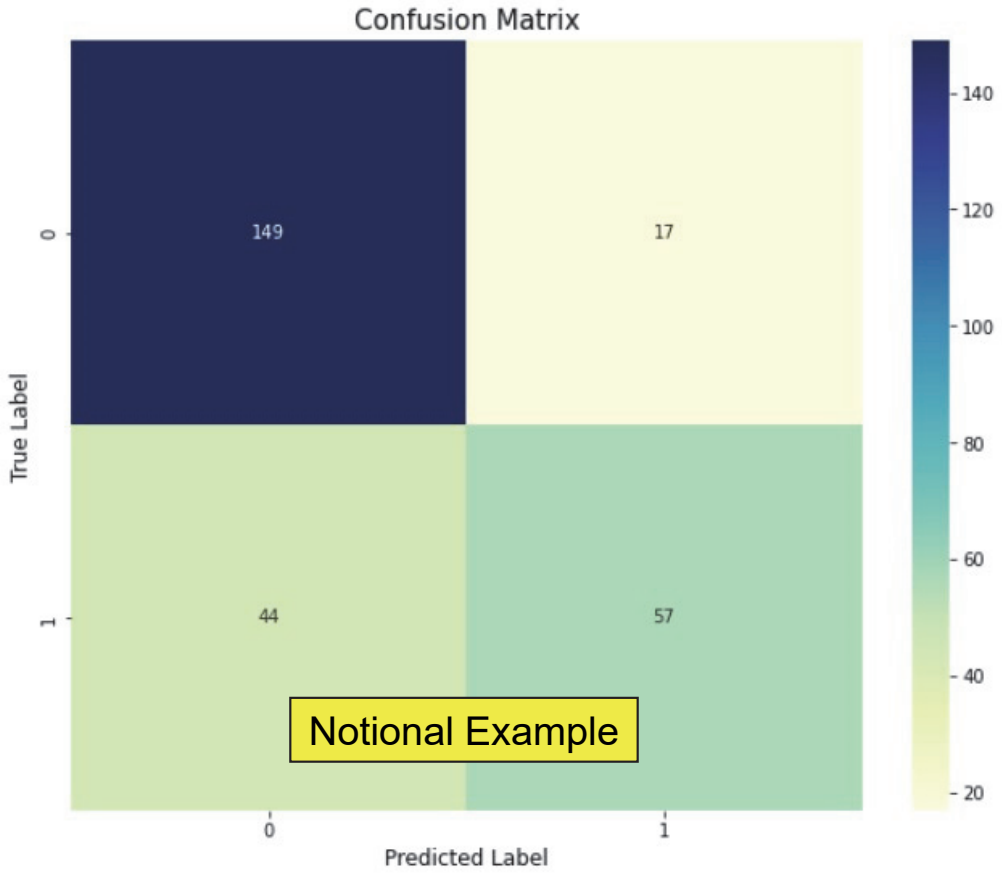
Evaluating a classification model

- For this course, our main criteria will be classification *accuracy* $:= \frac{correct\ classifications}{total\ observations}$
  - As expected, we want to focus on accuracy with respect to a test set.
  - Hyperparameter tuning and variable selection via cross-validation with accuracy as the scoring criteria.

- We consider $Y = 1$ a positive and $Y = 0$ a negative.
- Additional Metrics:
  - True Positive Rate:= $TPR = \frac{correct\ Y=1\ classifications}{total\ Y=1\ observations}$
  - True Negative Rate:= $TNR = \frac{correct\ Y=0\ classifications}{total\ Y=0\ observations}$
  - True Positive Rate:= $FPR = \frac{incorrect\ Y=1\ classifications}{total\ Y=0\ observations}$
  - True Negative Rate:= $FNR = \frac{incorrect\ Y=0\ classifications}{total\ Y=1\ observations}$

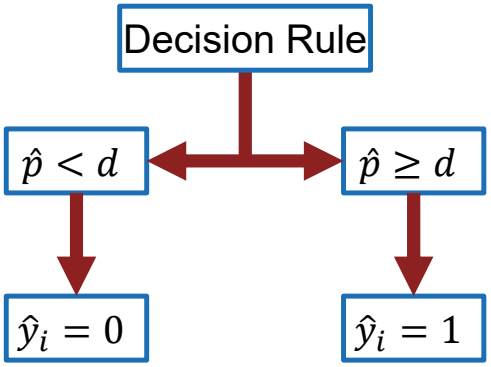
A Confusion Matrix summarizes performance data

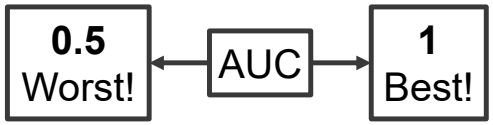
- When moving onto your test data, you can score your model using accuracy.
- After determining your model's classification predictions, you can produce a confusion matrix.
- The Confusion Matrix contains all the information needed to calculate:
  1. Accuracy
  2. True Positive Rate
  3. True Negative Rate
  4. False Positive Rate
  5. False Negative Rate

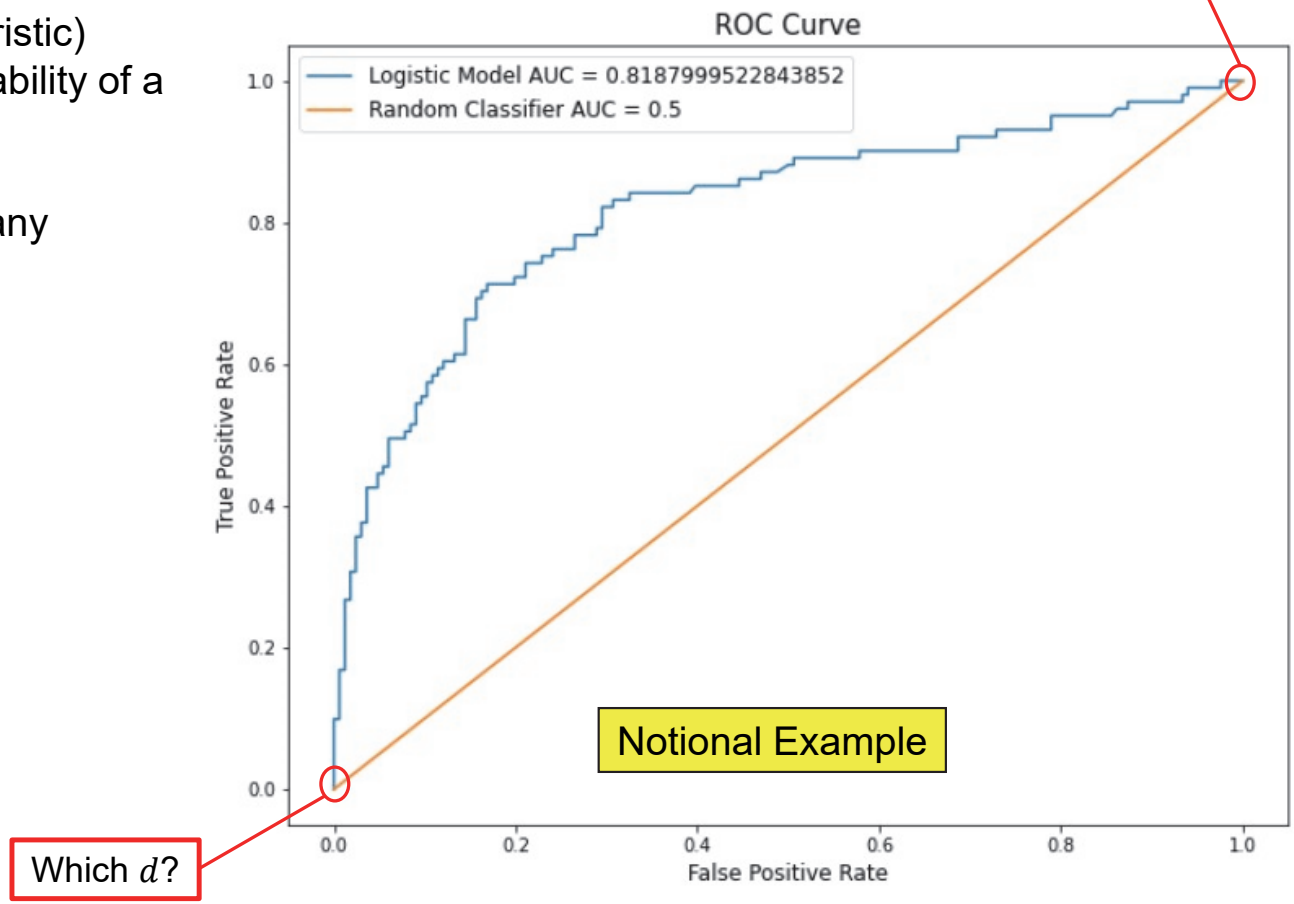


ROC Plots and AUC statistics are used to compare classifiers

- ROC (Receiver operating characteristic) curves show us the diagnostic capability of a binary classifier.
- Shows TPR against FPR across many different values of d where:







AUC: Area Under Curve; FPR: False Positive Rate; ROC: Receiver Operating Characteristic; TPR: True Positive Rate

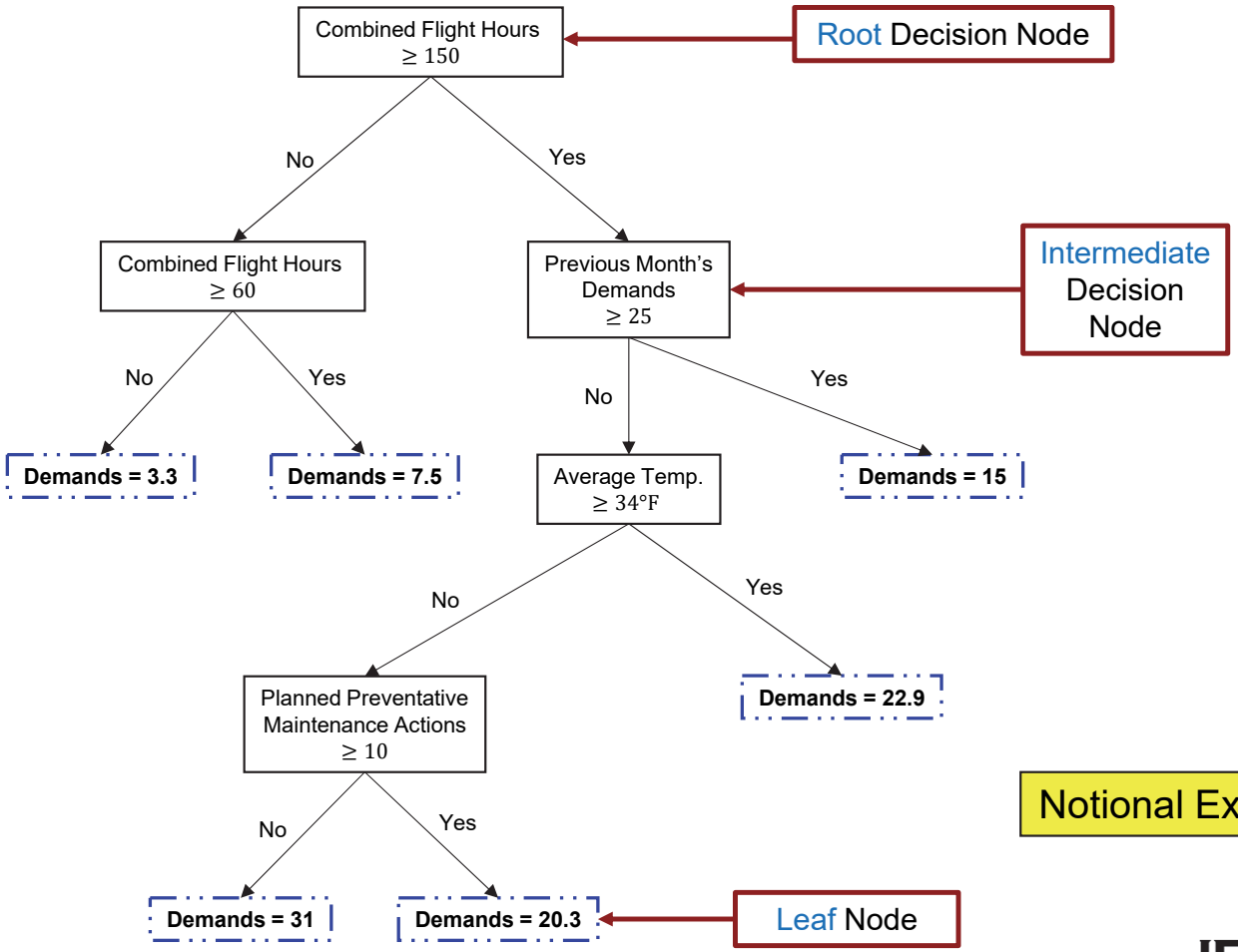
A quick overview of Decision Trees

The idea behind decision trees is simple

Decision nodes should be ordered based on what features seem the most important to making a correct decision.

(ex) Rain seems to be the most important indicator of whether there will be traffic or not.

Starting at the root, can we make a regression prediction or classification decision based on a series of sequential decisions?



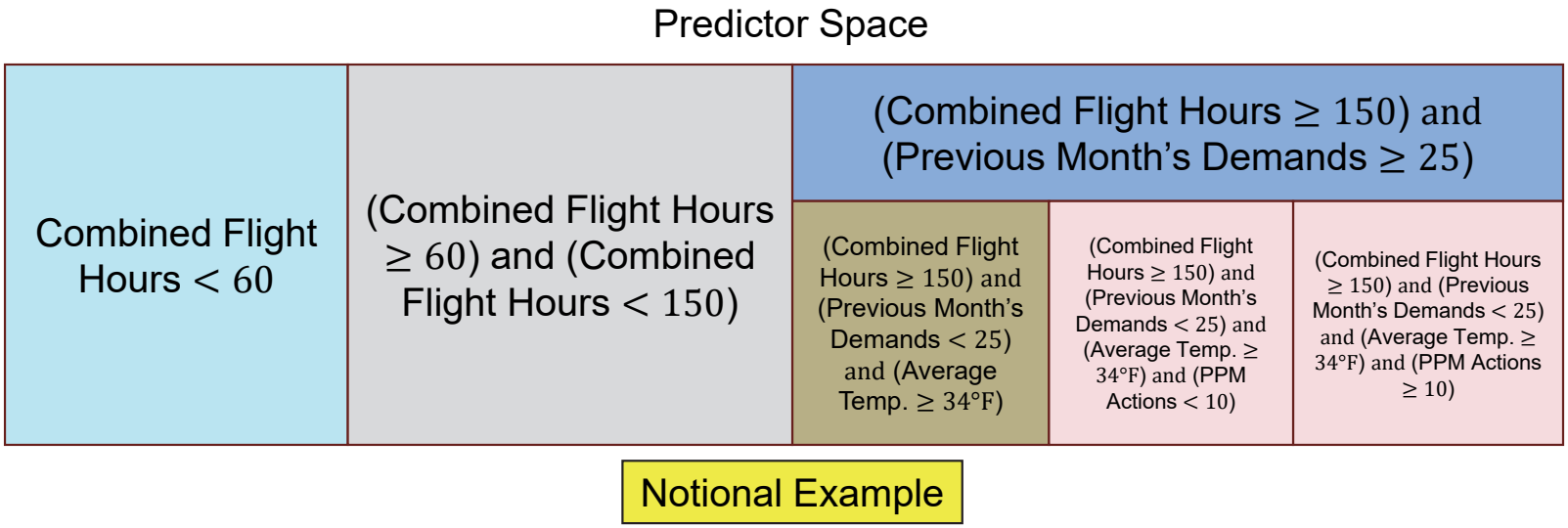
Notional Example

Decision Trees apply in regression and classification contexts

Bottom line: We *partitioned* our feature space into a number of disjoint (non-overlapping) regions called leaf nodes (or terminal nodes).

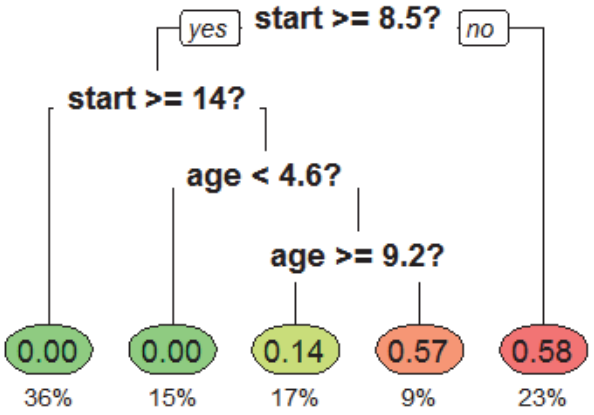
When a new test case arrives, the prediction is the most likely outcome in the region that this test case falls into:

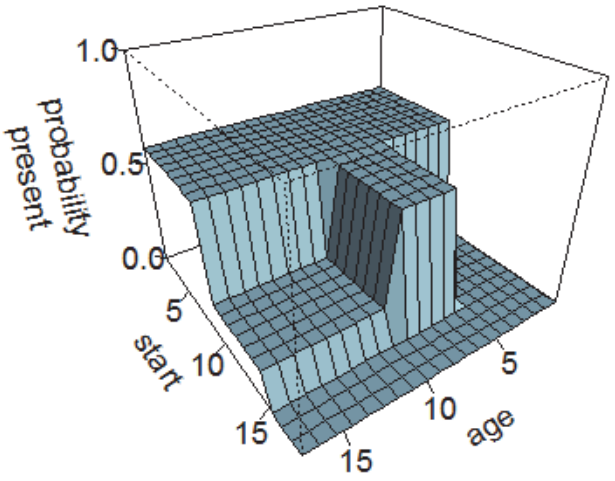
- Regression: Region Average,
- Classification: Region's most prevalent class

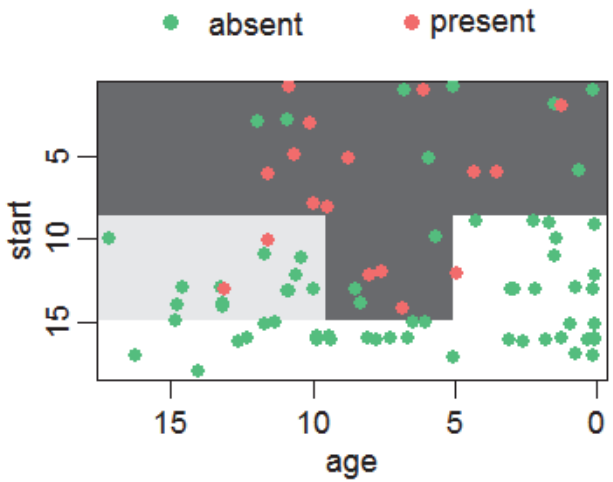


We can model nonlinear behavior easily with decision trees

The below decision tree nonlinearly models the probability of non-adults contracting a disease using averages of a binomial response variable.







Building even a single decision tree is a nontrivial problem and is usually done via heuristic methods such as *binary splitting*.

Ensembles of decision trees are some of the most powerful techniques

- Bagging, Random Forests, and Boosting methods use decision trees as building blocks to construct more powerful prediction models.
    a) Some of the best-performing machine learning techniques as of today.
    b) Improved predictive ability comes at the cost of easy interpretation!

- All of these methods construct many trees to improve performance. This is motivated by the fact that single trees have high variance.
- Both Bagging and Random Forests build trees by sampling from the original data using a technique called the Bootstrap.
- Boosting methods sequentially build decision trees that correct for errors in the previous trees.

Ensembles of trees do come at a cost of interpretability in exchange for their high performance

Note that ensembles of classification and regression trees (including Random Forests) are essentially of the form:

$$f(x) = \sum_{k=1}^{M} w_k \phi_k(x)$$

Boosting is a greedy algorithm for fitting weak learners (such as shallow trees) sequentially by fitting subsequent learners to adjust for the errors in the prior learner(s).

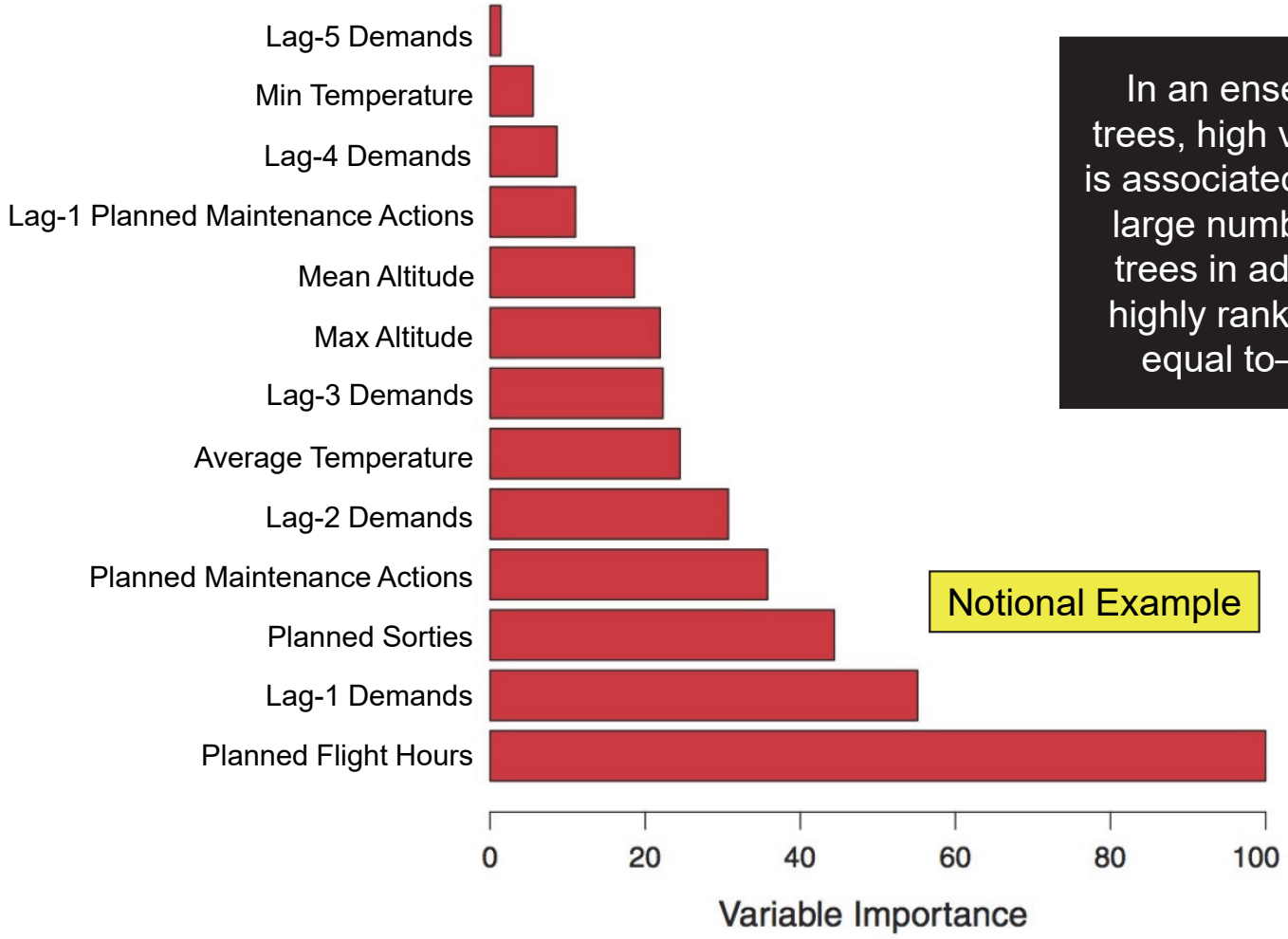
The best off-the-shelf performers on tabular data tasks are often boosted tree ensembles, such as *XGBoost*.

Variable Importance Plots are also useful for boosted ensemble interpretability. Note that, since these ensembles usually consist of hundreds or thousands of trees, interpretability takes a hit.

There is still much work on development of individual trees themselves. Recall that ***binary splitting*** is a rough heuristic for growing trees. A good place to start is the Interpretable Machine Learning Lab at Duke University run by PI Cynthia Rudin.

- Optimal Sparse Decision Trees (AISTATS, 2024, R. Zhang, R. Xin, M. Seltzer, and C. Rudin)
- Optimal Sparse Regression Trees (AAAI, 2023, R. Zhang, R. Xin, M. Seltzer, C. Rudin)
- Fast Sparse Decision Tree Optimization via Reference Ensembles (AAAI, 2022, H. McTavish, C. Zhong, R. Achermann, I. Karimalis, J. Chen, C. Rudin, M. Seltzer)

Sample Variable Importance Plot (VIP)



In an ensemble of decision trees, high variable importance is associated with presence in a large number of the decision trees in addition to it being a highly ranked split (close—or equal to—the root node).

Going Forward

Personally, I like to consider multiple models when performing a task

- Images or Free Text → NNs such as transformers
- Tabular data → XGBoost or CatBoost (if accuracy is paramount)
- Working with SMEs → Interpretability from decision trees or linear models also helps understand data and diagnose errors
- Large feature space but a need for interpretability → Regularization

Ultimately, try multiple models and compare! Even computer vision tasks can succeed quite well with interpretable methods if we do some combination of unsupervised restructuring prior to the supervised learning task.

NLP: Natural Language Processing; NN: Neural Network; SME: Subject Matter Expert

General optimization problem form for deriving machine learning models, one can propose their own

$$\begin{aligned} &\min && L(\theta|X,y,\lambda) \\ &subject\ to && f_1(\theta|X,y,\lambda) \geq 0 \\ & && \vdots \quad\quad \vdots \\ & && f_m(\theta|X,y,\lambda) \geq 0 \end{aligned}$$

- $L(\vec{\theta}|X,y,\lambda)$: is a loss function dependent on given "training" data (X,y) with decision variable vector θ and hyperparameter vector λ. Seeks to capture some goal.
- θ: a vector of multiple parameters (decision variables, relative to the optimization problem).
- λ: a vector of *hyperparameters* that are *fixed* relative to the optimization. Tuning these will be an "outer" optimization.
- Some constraints may exist on the decision variables (unconstrained is commonplace too).
- Parameters formulate a decision rule and generally provide some form of model interpretability.
- Note: X, y, λ are just fixed values relative to the optimization problem.

Recommended Further Reading and Packages

- *An Introduction to Statistical Learning* (with Applications in R/Python) by James, Witten, Hastie, and Tibshirani is available for free on statlearning.com
- A follow-up text is *The Elements of Statistical Learning* by Hastie, Tibshirani, and Friedman, which is available for free on hastie.su.domains/ElemStatLearn/
- For tasks involving tabular data, XGBoost and CatBoost are some of the most successful packages, regularly outperforming neural networks on tabular data. XGBoost is available on many platforms: xgboost.readthedoc.io provides excellent documentation from installation to implementation with examples. catboost.ai provides similar documentation. Both packages are open-source.
- Cynthia Rudin's Interpretable Machine Learning Lab provides many new approaches to growing and pruning decision trees.
- Note: for nontabular data such as free text (natural language processing) and images (computer vision), neural network architectures, such as transformers, will generally outperform anything done by decision trees—even vectorizing images for tabularization does not help as neural networks often leverage structures such as convolutions and attention mechanisms that extract features unique to images and text.

Final Questions?

Backup Slides

Optimization Problems

A general optimization problem is of the form:

$$\begin{aligned} \min \quad & f(x_1, x_2, \ldots, x_p) && \leftarrow \text{Objective Function} \\ subject\ to \quad & g_1(x_1, x_2, \ldots, x_p) \geq 0 && \leftarrow \text{First constraint} \\ & \vdots && \vdots \\ & g_n(x_1, x_2, \ldots, x_p) \geq 0 && \leftarrow \text{n}^{\text{th}} \text{ constraint} \end{aligned}$$

$x_1, x_2, \ldots, x_n$ are called the decision variables.

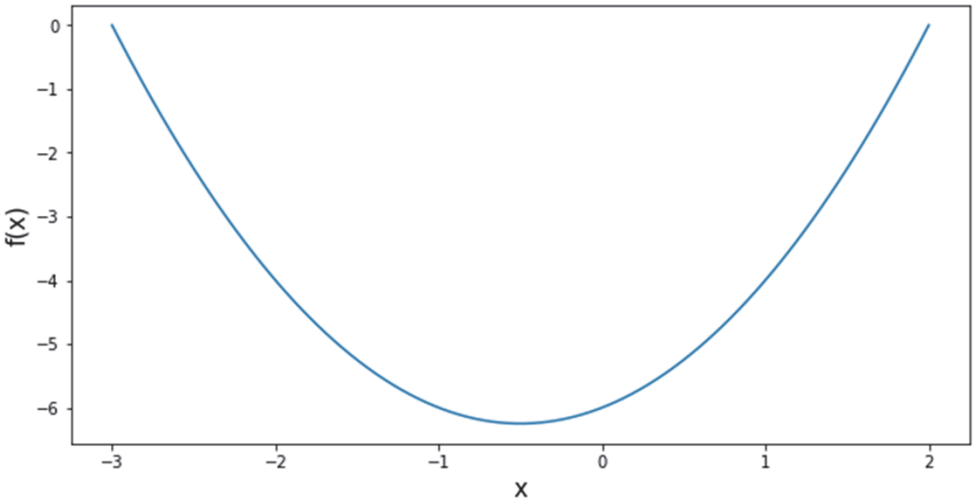
Simple algebra example:

$$\min \quad x^2+x-6$$

No constraints in this formulation!

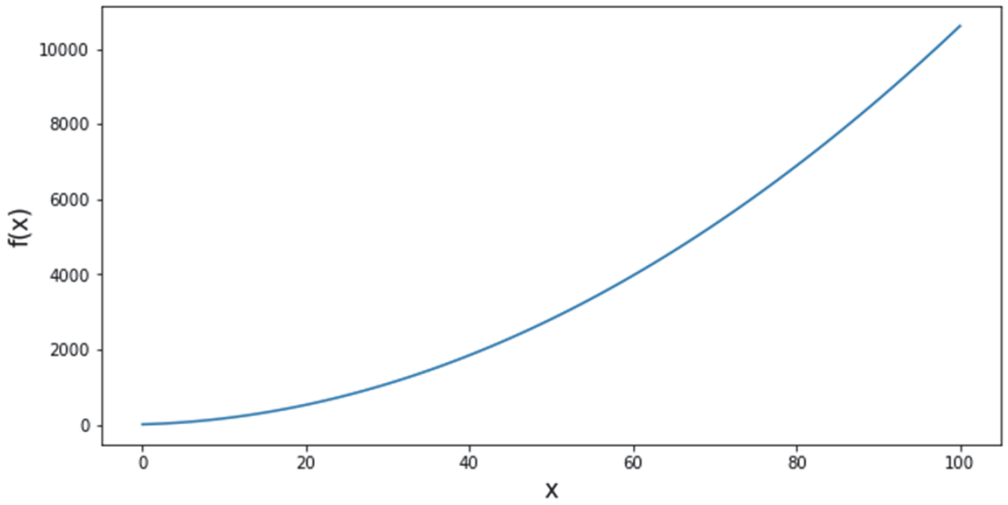
What about:

$$\begin{aligned} \min \quad & x^2+6x+9 \\ subject\ to \quad & x \geq 0 \end{aligned}$$



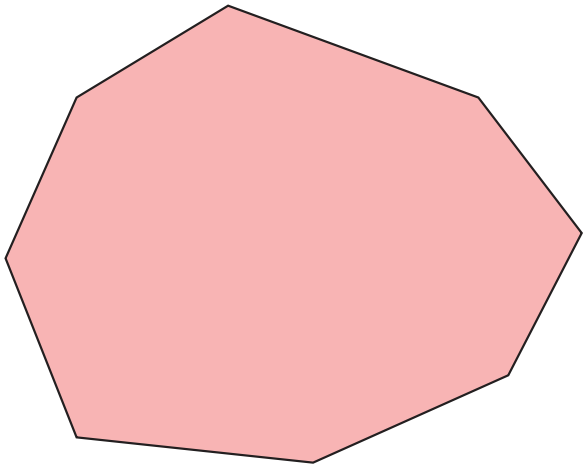
Unconstrained optimization: no boundaries to worry about.

Constrained optimization: optimal solution may lie on the boundary.

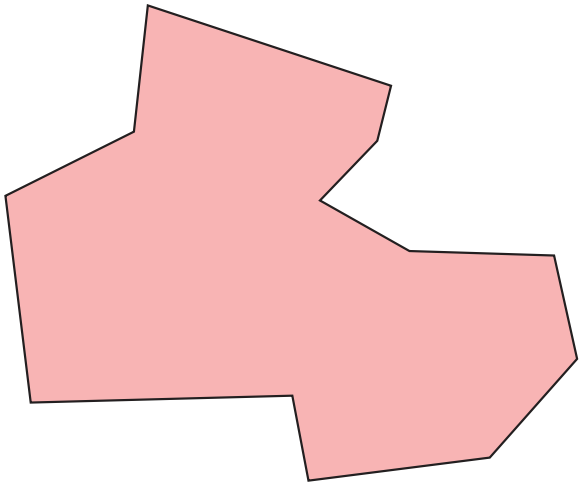


Convexity of Feasible Regions

Convex



Nonconvex



Which type of region is easier to navigate?

Optimizing

General rule of thumb: nonconvexity makes it very hard to find a global minimum.

Convex Optimization (many solved via interior-point or gradient-descent methods):

- Linear Programs
- Quadratic Programs
- Second-order Conic Programs
- Semi-definite Programs

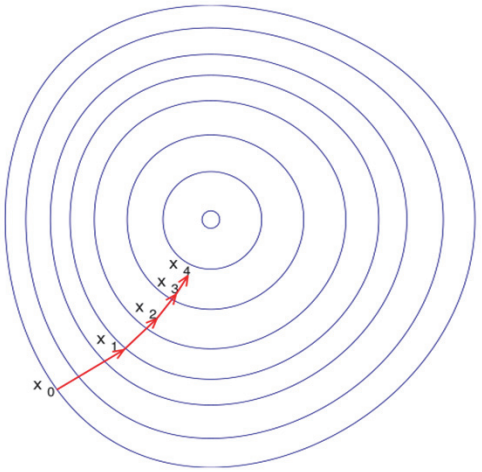
Nonconvex Optimization (usually no optimality guarantee):

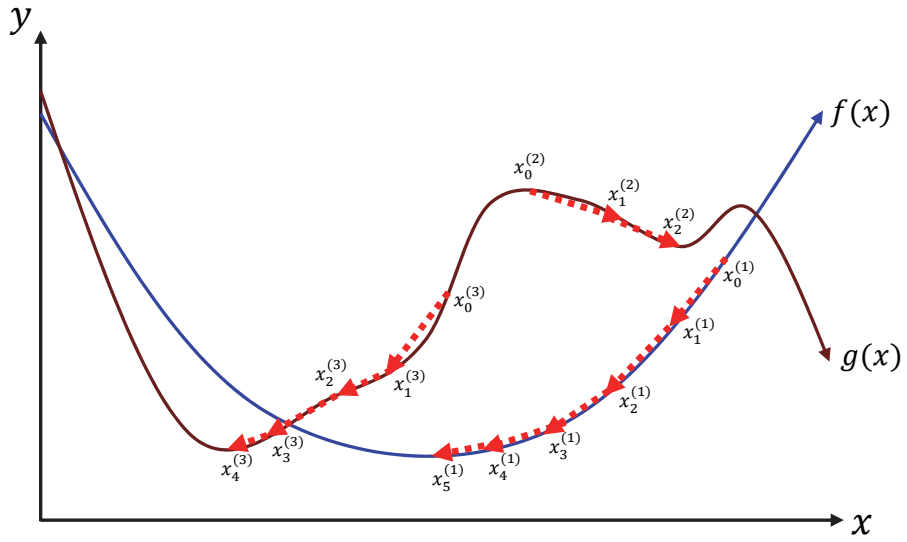
- Quasi-convex functions (can usually get global minimum)
- Difference of Convex functions (can get global minimum, depending on structure)
- Continuous nonconvex
- Combinatorial Optimization (the hardest?)

Gradient Descent is most popular for unconstrained optimization problems, regardless of convexity:

1. Pick a starting spot.
2. Find the direction of the steepest slope (gradient) and move a bit in that direction.
3. Repeat 2 until you have stopped improving (under some kind of stopping criteria).

Level set view of a 2-dimensional function

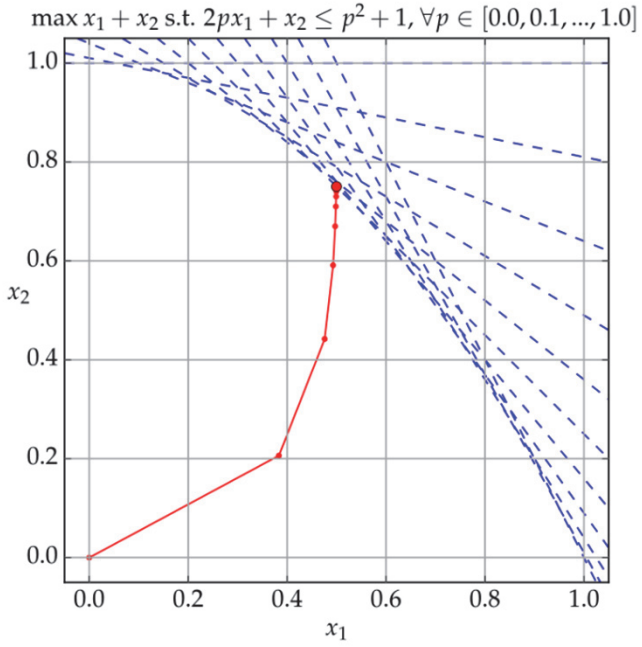




- Convex optimization: can get arbitrarily close to global minimum in a finite number of steps.
- Nonconvex optimization: can only guarantee a local minimum in a finite number of steps.
  - Local minimum might still be very good (Deep Learning).

Interior-point Methods are the most popular technique for constrained, convex optimization.

1. Start on the interior of the convex feasible region.
2. Mimic unconstrained optimization by encoding constraints in the objective with “barrier functions”.
3. Follow a path-finding algorithm (generally gradient-based) to reach the minimum.



Regression Tree Loss Function: (given a fixed J)

$$\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j} \right)^2$$

This entails identifying the partition that minimizes the RSS, where $\hat{y}_{R_j}$ is the mean response for the training observations within the j-th box.

It is computationally infeasible to consider every partition of J regions. Instead, we use a greedy and recursive approach called binary splitting.

Binary splitting; greedy, top-down approach:

- Select predictor X_j and cutpoint s such that splitting the predictor space into the regions $\{X \mid X_j < s\}$ and $\{X \mid X_j \geq s\}$ leads to the greatest possible reduction in RSS.

  That is, for any j and s, we consider the half-planes:

$$R_1(j,s) = \{X \mid X_j < s\} \text{ and } \mathrm{R}_2(\mathrm{j},\mathrm{s}) = \{X \mid X_j \geq s\}$$

  Then, we seek the value of j and s that minimize the equation:

$$\sum_{i:x_i \in R_1(j,s)} \left(y_i - \hat{y}_{R_1}\right)^2 + \sum_{i:x_i \in R_2(j,s)} \left(y_i - \hat{y}_{R_2}\right)^2$$

- Repeat the above procedure until a pre-specified stopping criterion is met.

- Binary splitting will perform well on the training data; however, it is prone to overfitting—especially when the number of splits, J is large.

- How do we choose a simpler tree (which boosts interpretability) that still performs well?

- Idea: require that each RSS reduction exceed a high enough threshold; however, this doesn't work well in practice as often a "meaningless" cut is often followed by a much better split.

- Better Idea: build a large decision tree T_0 (e.g., until no region has more than 5 observations) and then prune the tree to find the "best" subtree T via weakest link pruning.

Rather than considering every possible subtree, we consider a *sequence* of trees indexed by a tuning parameter $\alpha \geq 0$.

For each α, there is a subtree $T \subset T_0$ that minimizes:

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha|T|$$

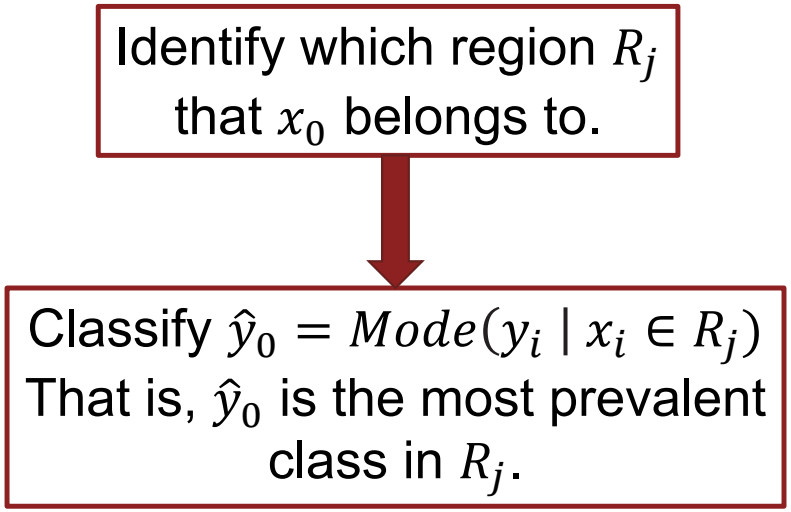
where $|T| =$ the number of terminal nodes in T.

- α acts as a penalty for the number of terminal nodes in a tree; i.e., it penalizes the complexity of the tree.
- $\alpha = 0 \rightarrow \mathrm{T} = \mathrm{T}_0$.
- Recall that T_0 results from our binary splitting algorithm until all terminal nodes have less observations than a specified number.

Classification Trees

Suppose each y_i belongs to one of K classes; i.e., $y_i \in \{1, \ldots, K\}$.

If we receive a new observation x_0, we classify according to the following rule:



- A partition of the feature space into regions $R_1, \ldots, R_J$ can again be chosen via binary splitting; however, regions can have more than one class in them, so we need a *measure* of the *class mixture* of a region.

- $\hat{p}_{j,k} :=$ the proportion of training observations in region R_j from class k.

- Three common cost measures of a region's class mixture:

1. Classification Error Rate: the fraction of observations not belonging to the dominant class,

$$E_j = 1 - \max_{k \in \{1,\ldots,K\}} \left(\hat{p}_{j,k}\right) \in [0,1]$$

2. Gini Index: the total variance across the K classes,

$$G_j = \sum_{k=1}^{K} \hat{p}_{j,k}\left(1 - \hat{p}_{j,k}\right)$$

3. Cross-entropy: similar to what we've seen in logistic regression,

$$D_j = -\sum_{k=1}^{K} \hat{p}_{j,k} \log\left(\hat{p}_{j,k}\right)$$

- The goal for any of the three mixture costs is to minimize.

- A few things to note about mixture costs:
  1. When building a classification tree, one typically uses either the Gini Index or the Cross-entropy, since they are more sensitive to node purity than classification error rate.

  2. When *pruning* a tree, the classification error rate is generally used (via cross-validation).

  3. More precisely: for fixed α (or alternatively, the number of terminal nodes), at this point in the pruning, we find subtree $T \subset T_0$ that minimizes:

$$\sum_{m=1}^{|T|} \left(1 - \max_{k \in \{1,\ldots,K\}} \hat{p}_{m,k}\right) + \alpha|T|$$

Reducing Variance via Averaging

- Recall our earlier problem: Decision Trees suffer from High Variance.

- **Fact**: Let $X_1, \ldots, X_n$ be independent random variables such that

$$Var(X_i) = \sigma^2 \text{ for all } i,$$

Then:

$$Var\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{\sigma^2}{n}$$

- Key Takeaway: the average of n random variables has n-fold smaller variance than the variance of each random variable taken individually. Can we use this to reduce the variance of trees?

Bagging

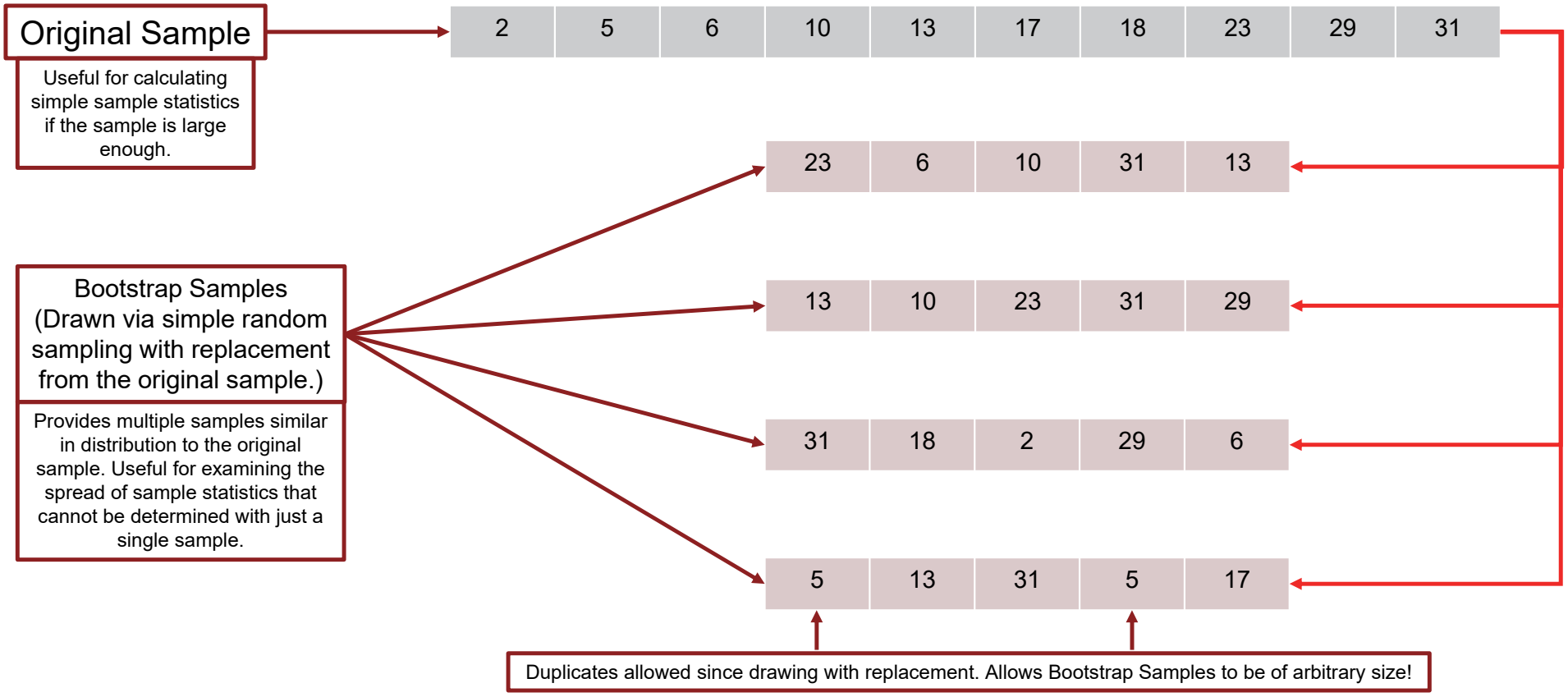
Bootstrap Aggregating

Idea:

- Create multiple bootstrap datasets from the training data.
- Fit a decision tree to each set.
- Average the results to reduce the variance.

The Bootstrap

- Basic Idea: resampling from a sample (with replacement).



Out-of-Bag Error Estimation

We can easily estimate the test error in a *Bagged Model*.

For a moderately sized dataset (or larger), the chance of a single observation being included in a given bootstrap sample is roughly 63%:

$$P(x_i \in bootstrap\ sample\ b) = 1 - \left(1 - \frac{1}{n}\right)^n \rightarrow 1 - \exp(-1)$$

$$\text{as } n \rightarrow \infty.$$

The remaining observations are not used to fit a given bagged tree and are called Out-of-Bag (OOB) observations.

- These can be viewed like the "subvalidation" sets used in cross-validation.
- These OOB observations can thus be used to estimate the test error.

Notes on Bagging:

When bagging, one typically grows "deep" trees for each bootstrap sample $i = 1, \ldots, B$ to ensure low bias. Naturally, these will be high-variance trees due to overfitting, but averaging reduces this large variance.

Computation time may be a consideration for "deepness" of trees since multiple trees must be fit. Usually an upper limit on the size of each tree (number of leaf nodes) is decided on a priori.

The value of B is not critical:

- Using a very large value for B will not lead to overfitting.
- As B increases, more computation is required.
- In practice: $B = 100$ to $B = 1000$ works pretty well.

Motivation for Random Forests:

Suppose there is one very strong predictor in the dataset along with a number of moderately strong ones.

In the collection of bagged trees, most or all of the trees will use this strong predictor in the top split; they will then proceed to all look somewhat similar.

This often leads to predictions from the bagged trees that are highly correlated.

Fact: averaging many highly correlated predictions does not lead to as large of a variance reduction as with uncorrelated predictions.

- Thus, bagging may not give a significant variance reduction compared to a single tree.

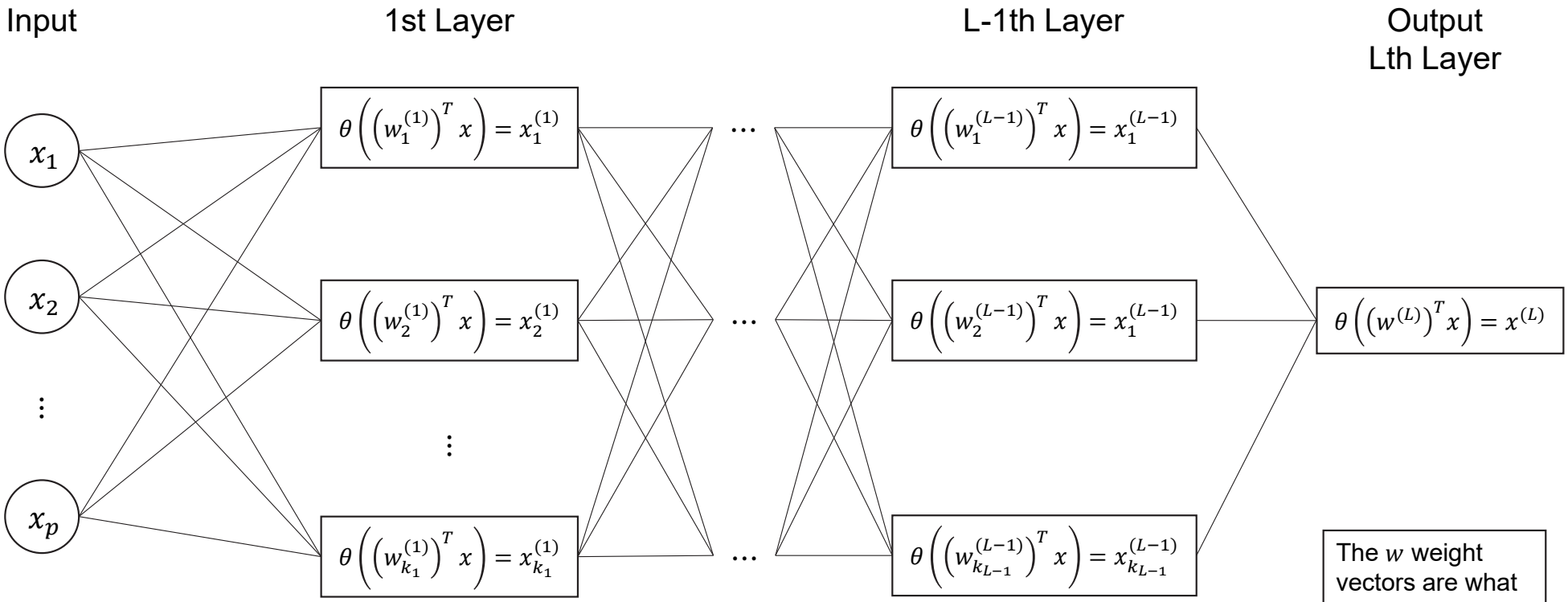
Random Forests

Goal: Remove correlation issues that result from Bagging.

Idea: For each bootstrapped sample $i = 1, \dots, B$, only allow a random selection of $m < p$ predictors for building tree T_i.

Result: Redundancies caused by "major players" in predictors are lost and the trees $T_1, \dots, T_B$ become "more independent".

Basic Neural Network Idea (Feedforward Network, *Multilayer Perceptron*)



Popular $\theta(x)$ choices: $\theta(x) = \frac{1}{1+e^{-x}}$, $\theta(x) = \tanh(x)$, $\theta(x) = \max(0, x)$

Like some other models (not mentioned explicitly), not limited to single output.

What's special about this architecture?

Linear models fed through nonlinear functions allow simple differentiation calculations for gradient descent. In general, however, loss functions are nonconvex.

Universal Approximation Theorems for Neural Networks show that specific types of neural networks are *excellent* approximators for general functions between two Euclidean spaces.[1,2,3]

Cons:

- The nonconvexity of the loss functions do not guarantee we are modeling the best function
- Overfitting is likely
- Usually loses to XGBoost on tabular data (operational data, economic data, social data)

Pros:

- NNs, with specific architectures, vastly outperform other methods when working with images and natural language data
- Excellent for generative AI; e.g., encoder-decoder networks

1: Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*. 2(4), pp. 303—314.

2: Hornik, K., Stinchcombe, M., White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), pp. 359—366.

3: Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks* 2(3), pp. 183—192.

Algebra Prerequisites

Matrices and Vectors

- Recall linear systems of equations:

$$\begin{aligned} 2x &+ y = 5 \\ -0.5x &+ 2y = 15 \end{aligned}$$

We get $x = -\frac{10}{9}$

and $y = \frac{65}{9}$

- Method of substitution:

- Gaussian Elimination:

- A system of equations like this

$$\begin{aligned} 2x &+ y = 5 \\ -0.5x &+ 2y = 15 \end{aligned}$$

- Can be represented in matrix form:

$$\begin{bmatrix} 2 & 1 \\ -0.5 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5 \\ 15 \end{bmatrix}$$

- Higher dimensional system (n equations, p variables):

$$\begin{aligned} a_{1,1}x_1 + \cdots + a_{1,p}x_p &= b_1 \\ \vdots \qquad\qquad\quad & \vdots \quad\; \vdots \\ a_{n,1}x_1 + \cdots + a_{n,p}x_p &= b_n \end{aligned}$$

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,p} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,p} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Data: Feature Matrix & Response Vector

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & & x_{1,p-2} & x_{1,p-1} & x_{1,p} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,p-2} & x_{2,p-1} & x_{2,p} \\ x_{3,1} & x_{3,2} & x_{3,3} & & x_{3,p-2} & x_{3,p-1} & x_{3,p} \\ & \vdots & & \ddots & & \vdots & \\ x_{n-2,1} & x_{n-2,2} & x_{n-2,3} & & x_{n-2,p-2} & x_{n-2,p-1} & x_{n-2,p} \\ x_{n-1,1} & x_{n-1,2} & x_{n-1,3} & \cdots & x_{n-1,p-2} & x_{n-1,p-1} & x_{n-1,p} \\ x_{n,1} & x_{n,2} & x_{n,3} & & x_{n,p-2} & x_{n,p-1} & x_{n,p} \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-2} \\ y_{n-1} \\ y_n \end{bmatrix}$$

Features/predictors are across the columns.
Observations are across the rows.
$x_{i,j}$ is the value of feature j for observation i.

Each observation has a response value.
Each response is paired with a row in X.
y_i is the response of observation i.

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,p} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,p} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

We can assign letters to matrices and write this equation as:

$$A\vec{x} = \vec{b}$$

- $A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,p} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,p} \end{bmatrix}$ is an $n \times p$ matrix.

- $\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$ and $\vec{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$ are both vectors with dimensions $\mathrm{p} \times 1$ and $\mathrm{n} \times 1$ respectively.

Vectors are a matrix where one of the dimensions (either row or column) is equal to 1:

Row vector: $\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$

Column Vector: $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

What's the difference? Aren't they just ordered lists?

Matrix dimensions matter for matrix multiplication

A is $n \times p$ and B is $m \times q$

You can only evaluate AB (standard matrix multiplication) if $p = m$.

Matrix multiplication is generally non-commutative: $AB \neq BA$.

- Some special matrices may be commutative.
- Due to the dimensional requirements of multiplication, AB might exist, but BA **does not!**

Let A be an $n \times k$ matrix and let B be a $k \times p$ matrix:

- AB exists and is an $n \times p$ matrix.
- BA only exists if $n = p$.

Multiplication with vectors: Suppose $\vec{x} = [x_1 \quad \cdots \quad x_p]$ and $\vec{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_p \end{bmatrix}$.

$$\vec{x}\vec{y} = x_1y_1 + x_2y_2 + \cdots + x_{p-1}y_{p-1} + x_py_p$$

Transpose of a matrix A: "diagonal rotation"

- Called A^T or "A transpose".
- The i-th column of A becomes the i-th row of A^T; equivalently, the i-th row of A becomes the i-th column of A^T.
- (ex)

$$A = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}, A^T = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$$

$$B = \begin{bmatrix} 5 & 9 & -2 \\ 3 & 0 & -1 \\ 11 & -9 & 3 \end{bmatrix}, B^T = \begin{bmatrix} 5 & 3 & 11 \\ 9 & 0 & -9 \\ -2 & -1 & 3 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 3 & 0 \\ 3 & 2 & -1 \\ 0 & -1 & -1 \end{bmatrix} = C^T = \begin{bmatrix} 1 & 3 & 0 \\ 3 & 2 & -1 \\ 0 & -1 & -1 \end{bmatrix}$$

Let A and B be matrices such that AB exists. This product is calculated as follows:

- The value for row-i, column-j in AB is the product of the i-th row of A with the j-th column of B.
- (ex)

$$A = \begin{bmatrix} 1 & 0 & -1 \\ 2 & -1 & 3 \\ 4 & -4 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} -1 & 2 & 0 \\ 0 & -1 & 1 \\ -2 & 3 & 2 \end{bmatrix}$$

$$AB = \begin{bmatrix} 1 & -1 & -2 \\ -8 & 14 & 5 \\ -4 & 12 & -4 \end{bmatrix}$$

- Warning! There is a matrix product called the "Hadamard Product" which simply multiplies element-wise (this product only works for matrices of the same dimension). This is far less common than the standard matrix product, but it appears in some applications like constrained matrix factorization and image processing.

Matrix addition/subtraction is simple, you simply add/subtract element-wise.

- (ex)

$$A = \begin{bmatrix} 1 & 0 & -1 \\ 2 & -1 & 3 \\ 4 & -4 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} -1 & 2 & 0 \\ 0 & -1 & 1 \\ -2 & 3 & 2 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 0 & 2 & -1 \\ 2 & -2 & 4 \\ 2 & -1 & 2 \end{bmatrix}$$

$$A - B = \begin{bmatrix} 2 & -2 & -1 \\ 2 & 0 & 2 \\ 6 & -7 & -2 \end{bmatrix}$$

Matrix Inversion

- Critical concept in data science.

- Recall for linear systems of equations, we can have:
  1. Unique solution
  2. Infinitely many solutions
  3. No solution

- Situation 1 most interests us:
  - Requires the same number of equations as variables, so the matrix representation involves a square matrix.
  - Let $A\vec{x} = \vec{b}$ be our linear system, where A is an $n \times n$ matrix.
    - If this was not involving matrices; i.e., ax = b, how would we solve for x?
    - This is not a Hadamard Product, so the inverse operation is not immediately apparent.

- If $A\vec{x} = \vec{b}$ has a unique solution, then there exists a matrix A^{-1} such that we can left multiply the system as follows:

$$A^{-1}A\vec{x} = \vec{x} = A^{-1}\vec{b}$$

  - Note: $A^{-1}A = I_n$ which is a square $n \times n$ matrix with all 0s except for all 1s on the main diagonal called an identity matrix.
  - (ex)

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  - Any matrix or vector left/right multiplied by an identity matrix (of the requisite dimensions) does not change: $I_n\vec{x} = \vec{x}$ and $I_nA = AI_n = A$.

- Not all square matrices are invertible.
  - We call uninvertible matrices singular.
  - Invertible matrices are nonsingular.
  - This occurs if at least one of the columns/rows can be obtained as a linear combination of the others; i.e., a sum of multiples of the other columns/rows.

inv(A) will compute the matrix inverse of A.†

Exercises:

$$\text{Let } A = \begin{bmatrix} 1 & 0 & -1 \\ 2 & -1 & 3 \\ 4 & -4 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} -1 & 3 & 0 \\ 0 & 0 & 1 \\ -2 & 6 & 2 \end{bmatrix}$$

1. Compute (or attempt to compute) the inverses of A and B. Did one fail? Why? For the one that worked, test that a matrix times its inverse gives the identity.

2. Compute the inverse of an inverse (for a nonsingular, well-conditioned matrix). What if it's ill-conditioned?

3. Replace the "6" in B with "5.999999999999999". Compute the inverse.
   What do you notice about it?
   Try replacing the "6" with "5" instead. What does the inverse look like now?

- Matrices that are technically nonsingular but are numerically close to being singular are called ill-conditioned matrices.

- Ill-conditioned matrices have inverses that "blow up" numerically.

- Let X be any real matrix.
  - Suppose X has linear dependent columns.
    - Guaranteed if there are more columns than rows.
  - Then $X^T X$ is singular.
    - May be computed as ill-conditioned due to numerical rounding on a computer.

- If you have more variables than observations, this can pose issues with some machine learning models which compute $(X^T X)^{-1}$!

- Redundant variables will lead to columns being linearly dependent or close to it!
  - (ex) Same measurement but different units, correlated indicators, etc.
  - Creates an issue called "multicollinearity".
  - Be careful when gathering data!
  - Note: we can always remove variables to treat this issue → Variable Selection.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
XX-04-2025	Final	

4. TITLE AND SUBTITLE

DATAWorks 2025: Interpretable Machine Learning 101

5a. CONTRACT NUMBER

HQ0034-24-D-0020

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Nikolai D Lipscomb

5d. PROJECT NUMBER

C9082

5e. TASK NUMBER

C9082

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses
730 East Glebe Road
Alexandria, Virginia 22305

8. PERFORMING ORGANIZATION REPORT NUMBER

3004260

3004260

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

This OED Draft has not been approved by the sponsor for distribution and release. Reproduction or use of this material is not authorized without prior permission from the responsible IDA Division Director.

13. SUPPLEMENTARY NOTES

Project Leader: Rebecca M Medlin

14. ABSTRACT

This mini tutorial focuses on introducing the methods and concepts used in interpretable machine learning, particularly for applications that incorporate tabular operational data.

Much of the current AI focus is on generative AI, which is deeply rooted in uninterpretable neural networks; however, there is still substantial research in interpretable machine learning that regularly outperforms neural networks for classical machine learning tasks such as regression and classification.

This course will summarize supervised vs. unsupervised machine learning tasks as well as online vs offline learning settings. Classical machine learning models are introduced and used to motivate concepts such as bias-variance trade-off, hyperparameter tuning, and optimization algorithms/convergence. Last, a swift overview of recent work in interpretable machine learning will point the audience towards state-of-the-art advances.

15. SUBJECT TERMS

Machine Learning, Artificial Intelligence, Statistics, Optimization, Data Science

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Rebecca M Medlin
Unclassified	Unclassified	Unclassified	SAR	112	19b. TELEPHONE NUMBER *(Include area code)* 703-845-6731