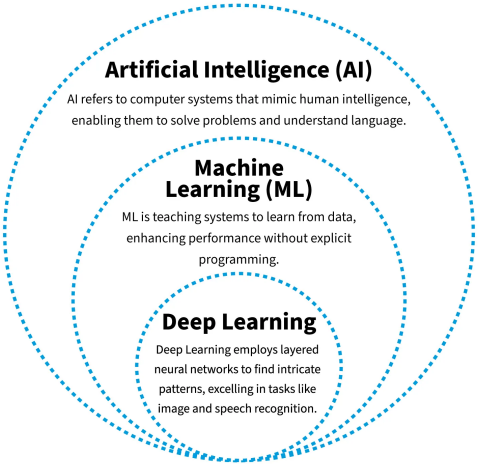
A Primer on ML/AI for population research

Tyler H. McCormick, Kentaro Hoffman
Univeristy of Washington

PAA 2025

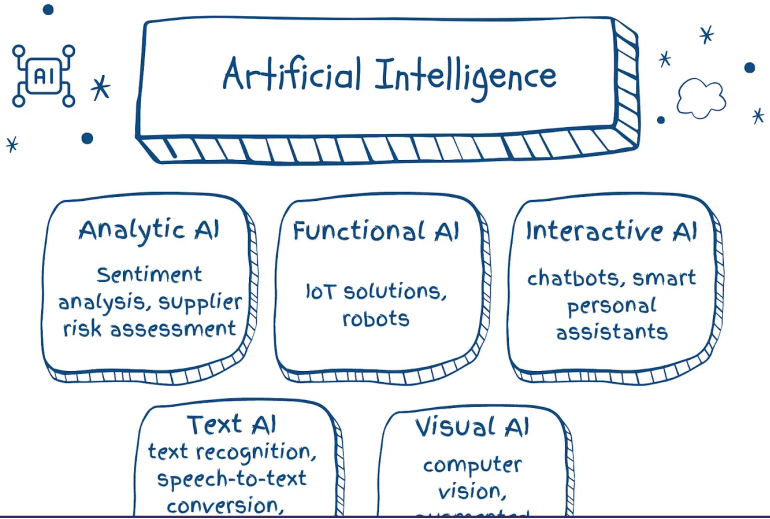
AI and ML



{https://www.interaction-design.org/literature/topics/ai}

AI is very broad.

{Source: https://medium.com/codex/defining-artificial-intelligence-machine-learning-and-data-science-b310119a76d1}



and intervene in human and non-human behavior. Generally speaking, machine learning refers to the practice of automating the discovery of rules and patterns from data, however dispersed and heterogeneous it may be, and drawing inferences from those patterns, without explicit programming.[1] Using examples drawn from social media, we

Figure 1: Loops, ladders, and links, Fourcade and Johns

Statistics, two cultures

Statistical Science
2001, Vol. 16, No. 3, 199–231

Statistical Modeling: The Two Cultures

Leo Breiman

Abstract. There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown. The statistical community has been committed to the almost exclusive use of data models. This commitment has led to irrelevant theory, questionable conclusions, and has kept statisticians from working on a large range of interesting current problems. Algorithmic modeling, both in theory and practice, has developed rapidly in fields outside statistics. It can be used both on large complex data sets and as a more accurate and informative alternative to data modeling on smaller data sets. If our goal as a field is to use data to solve problems, then we need to move away from exclusive dependence

Nature

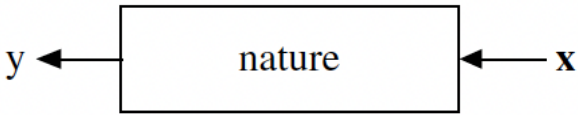
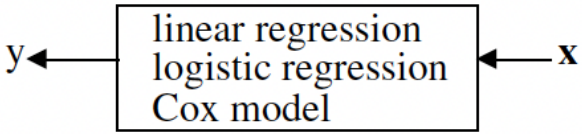


Figure 3: Statistics, two cultures

Data Modeling Culture

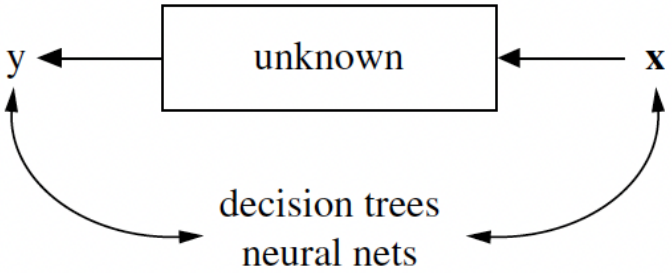
Assumes data are generated by a stochastic model.
Focus on parameter estimation and hypothesis testing.
Examples: Linear regression, logistic regression.



Model validation. Yes–no using goodness-of-fit tests and residual examination.
Estimated culture population. 98% of all statisticians.

Algorithmic Modeling Culture

Treats the data mechanism as unknown.
Focus on predictive accuracy.
Examples: Decision trees, neural networks.



Model validation. Measured by predictive accuracy.
Estimated culture population. 2% of statisticians, many in other fields

Prediction vs. Inference

Prediction: Forecasting future data points.
Inference: Understanding the underlying data mechanism.
Differences and applications.

Mathematical Details of Prediction and Inference

Prediction:

$$\hat{y} = f(x)$$

We care about estimating $\hat{y}$

Inference:

$$y = f(x) + \epsilon$$

where ϵ is the error term. We care about estimating $\hat{f}(\cdot)$.

Bias-Variance Trade-off

Definition: Trade-off between bias (error due to assumptions in the model) and variance (error due to variability in the model).

Importance: Balancing bias and variance is crucial for model accuracy.

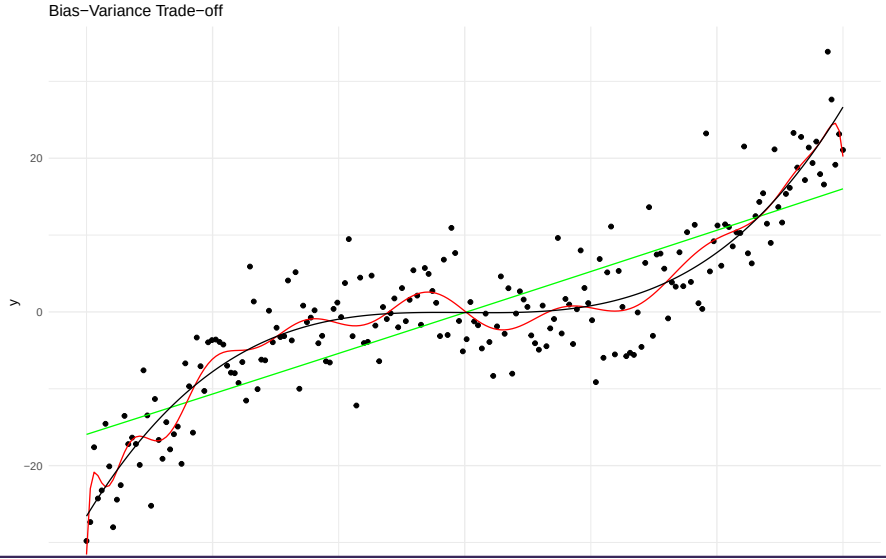
Bias-Variance Trade-off Example in R

```r
set.seed(123)
library(ggplot2)

# Generate sample data
n <- 200
x <- seq(-3, 3, length.out = n)
y <- x^3 + rnorm(n, sd = 5)

# Fit models with different complexities
model_low <- lm(y ~ poly(x, 2))
model_med <- lm(y ~ poly(x, 3))
model_high <- lm(y ~ poly(x, 20))
```

Bias-Variance Trade-off Example in R

```r
# Predictions
y_pred_low <- predict(model_low, data.frame(x = x))
y_pred_med <- predict(model_med, data.frame(x = x))
y_pred_high <- predict(model_high, data.frame(x = x))

# Plot
data <- data.frame(x = x, y = y, y_pred_low = y_pred_low,y_pred_med = y_pr
ggplot(data, aes(x = x, y = y)) +
  geom_point() +
  geom_line(aes(y = y_pred_low), color = 'green') +
  geom_line(aes(y = y_pred_high), color = 'red') +
  geom_line(aes(y = y_pred_med), color = 'black') +
  labs(title = 'Bias-Variance Trade-off', x = 'x', y = 'y') +
  theme_minimal()
```

Bias-Variance Trade-off Example in R



Explanation of R Example

Low complexity model (green line) has very high bias but very low variance.

Medium complexity model (black line) has lower bias but higher variance than the low complexity.

High complexity model (red line) has very low bias but very high variance.

Some additional things to consider

Rashomon Effect: Multiplicity of good models

Occam's Razor: Simplicity vs. accuracy

Bellman Effect: Curse of dimensionality

Rashomon Effect

The Rashomon Effect refers to the existence of multiple models that explain the data equally well.

In linear regression, many subsets of variables can provide similar predictive accuracy.

This leads to different models telling different stories about variable importance.

Occam's Razor

Occam's Razor advocates for simplicity in model selection.

Simpler models are preferred unless a more complex model significantly improves accuracy.

Occam's Razor (Contd.) Conflict

Simplicity vs. accuracy: More complex models often provide better predictions but are harder to interpret.

Bellman Effect

The Bellman Effect, also known as the curse of dimensionality, refers to the exponential increase in computational complexity as the number of dimensions increases.

Bellman Effect (Contd.) Implications

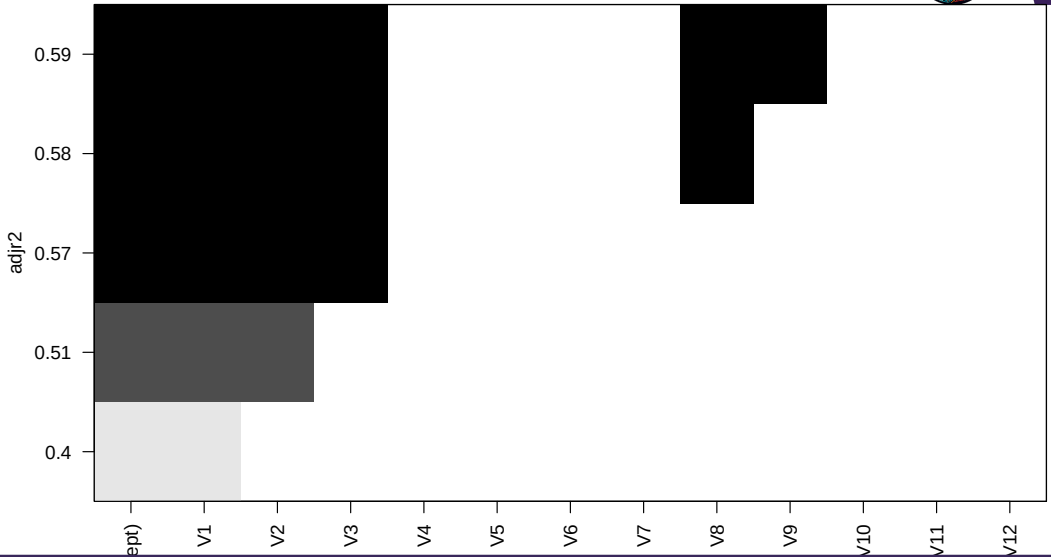
High-dimensional data can lead to overfitting and increased computational cost.

Techniques like dimensionality reduction are used to mitigate this effect.

Example in R - Rashomon Effect

```r
# Load necessary libraries
library(MASS)
library(leaps)

# Generate synthetic data
set.seed(123)
n <- 100
p <- 12
X <- matrix(rnorm(n * p), n, p)
beta <- c(1, 0.5, -0.5, rep(0, p - 3))
y <- X %*% beta + rnorm(n,3)

# Fit multiple linear regression models
models <- regsubsets(y ~ ., data = as.data.frame(X), nvmax = 5)
summary(models)
```

Example in R - Rashomon Effect



Definition of Supervised Learning

Supervised Learning is a type of machine learning where the model is trained on a labeled dataset.

Each training example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal).

Key Concepts

Training Data: The dataset used to train the model, containing input-output pairs.

Labels: The output values associated with each input in the training data.

Model: The algorithm that learns from the training data to make predictions.

Loss Function: A function that measures the discrepancy between the predicted output and the actual output.

Optimization: The process of adjusting the model parameters to minimize the loss function.

Supervised Learning Process

Data Collection: Gather a labeled dataset.

Data Preprocessing: Clean and prepare the data for training.

Model Selection: Choose an appropriate algorithm for the task.

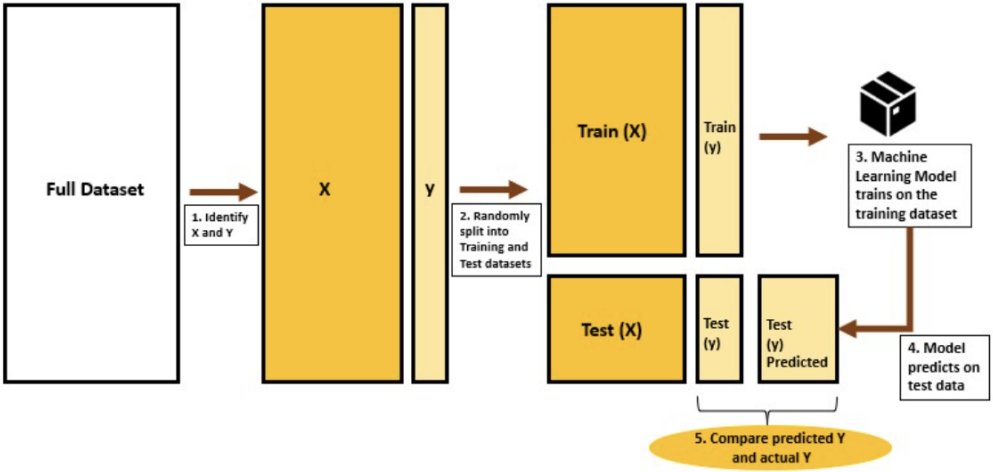
Training: Use the training data to fit the model.

Evaluation: Assess the model's performance on a separate validation set.

Prediction: Use the trained model to make predictions on new data.

Supervised learning

{Source: https://www.machinelearningplus.com/machine-learning/train-test-split/}



Key Differences with unsupervised learning

Data: Supervised learning uses labeled data, while unsupervised learning uses unlabeled data.

Objective: Supervised learning aims to predict outcomes, while unsupervised learning aims to find hidden patterns or intrinsic structures.

Algorithms: Common supervised learning algorithms include regression and classification, while unsupervised learning algorithms include clustering and dimensionality reduction.

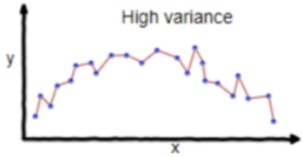
Regularized regression

Regularized Regression introduces a penalty term to the loss function to prevent overfitting.

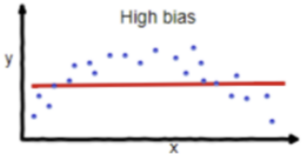
Common methods: **Lasso** (L1 regularization), **Ridge Regression** (L2 regularization), and **Elastic Net** (combination of L1 and L2).

Overfitting

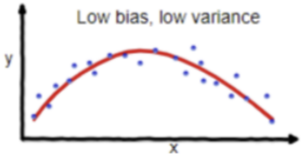
{Source: https://www.geeksforgeeks.org/lasso-vs-ridge-vs-elastic-net-ml/}



overfitting



underfitting



Good balance

Formulation

Ordinary Least Squares (OLS):

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2$$

Ridge Regression:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Lasso:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Elastic Net:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2$$

Likelihood and Computation

Likelihood:

$$L(\beta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{x}_i^T\beta)^2}{2\sigma^2}\right)$$

Objective Function:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T\beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Computation: Typically solved using **Coordinate Descent** or **Proximal Gradient Methods**.

Tuning Parameter

Selection: Chosen via **Cross-Validation** (e.g., Grid Search, Random Search, Bayesian Optimization).

Implications:

Large λ: More regularization, simpler model, higher bias.
Small λ: Less regularization, more complex model, higher variance.

Intuitive Explanation

Lasso: Encourages sparsity, many coefficients shrink to zero.

Ridge: Shrinks coefficients but does not set them to zero, useful when all predictors are believed to be relevant.

Elastic Net: Combines the benefits of both Lasso and Ridge, useful for highly correlated predictors.

Regularization

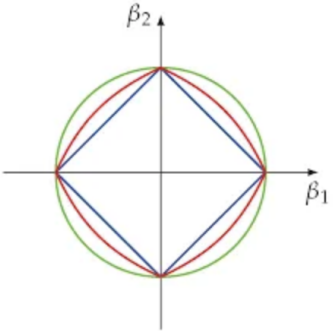


Figure 21: The Elastic Net (red) is a combination of ridge regression (green) and the lasso (blue). Image Citation: https://www.researchgate.net/figure/Visualization-of-the-elastic-net-regularization-red-combining-the-L2-norm-green-of_fig6_330380054

Least Angle Regression (LARS)

Definition: LARS is an efficient algorithm for fitting Lasso models.
Key Idea: LARS incrementally builds the model by adding predictors that are most correlated with the residuals.
Steps:

Start with all coefficients set to zero.
Identify the predictor most correlated with the response.
Move the coefficient of this predictor towards its least squares value until another predictor is equally correlated with the residual. _ Continue in this manner, adjusting coefficients along the equiangular direction until all predictors are included or the desired sparsity is achieved.

Advantages

Efficiency: LARS is computationally efficient, especially when the number of predictors is large.

Flexibility: Can be modified to produce the Lasso solution by incorporating a soft thresholding step.

Practical Use

Coefficient Path: LARS provides a coefficient path that shows how coefficients enter and leave the model as the regularization parameter varies.

Model Selection: Allows for easy selection of the optimal model based on desired sparsity or cross-validation.

Visualization: The coefficient path plot helps in visualizing the trade-off between model complexity and prediction accuracy.

Coefficient Path

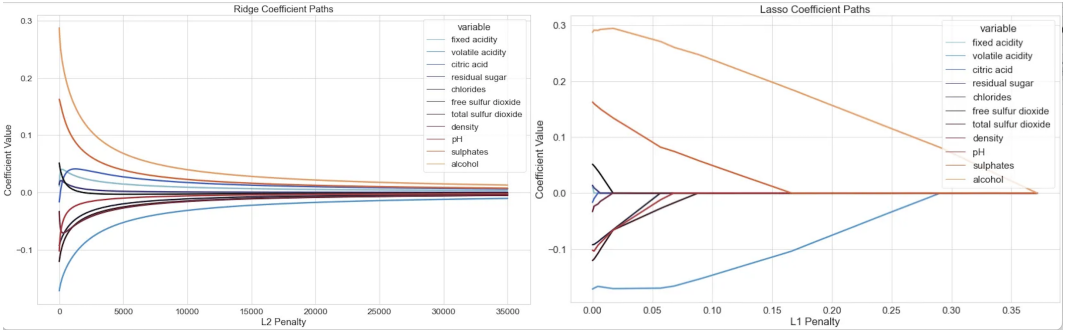


Figure 5: https://towardsdatascience.com/regularized-linear-regression-models-dcf5aa662ab9

Cross-Validation

Definition: A technique to assess the generalizability of a model by partitioning the data into training and validation sets.

Types:

K-Fold Cross-Validation: Divides the data into K subsets, trains the model K times, each time using a different subset as the validation set.
Leave-One-Out Cross-Validation (LOOCV): Uses a single observation as the validation set and the remaining observations as the training set.

Purpose: Helps in selecting the optimal model and tuning parameters by evaluating performance on unseen data.

Information Criteria to choose regularization

AIC (Akaike Information Criterion): Balances model fit and complexity.

Formula:

$$AIC = 2k - 2\ln(L)$$

where k is the number of parameters and L is the likelihood.

BIC (Bayesian Information Criterion): Similar to AIC but with a stronger penalty for complexity.

Formula:

$$BIC = \ln(n)k - 2\ln(L)$$

where n is the number of observations.

Practical considerations: Importance of Feature Scalin

Definition: Standardizing features to have zero mean and unit variance.

Purpose: Ensures that all features contribute equally to the regularization term.

Impact: Without scaling, features with larger scales dominate the penalty term, leading to biased coefficient estimates.

Methods of Feature Scaling

Standardization: Subtract the mean and divide by the standard deviation.

Formula:

$$x' = \frac{x - \mu}{\sigma}$$

Normalization: Scale features to a range, typically [0, 1].

Formula:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Lasso and multicolinearity: the irrepresentability Condi

Definition: A condition that determines whether Lasso can correctly identify the true model.

Mathematical Formulation: For Lasso to select the correct model, the following condition must hold:

$$\|\mathbf{X}_{-S}^T \mathbf{X}_S (\mathbf{X}_S^T \mathbf{X}_S)^{-1} \text{sign}(\beta_S)\|_\infty < 1$$

where S is the set of true non-zero coefficients, $\mathbf{X}_S$ is the design matrix for the predictors in S, and $\mathbf{X}_{-S}$ is the design matrix for the remaining predictors.

Implications for Covariate Selection

Implications: If the irrepresentability condition is violated, Lasso may fail to select the correct set of covariates.

Highly Correlated Predictors: Lasso may arbitrarily select one predictor from a group of highly correlated predictors, leading to instability in variable selection.

Practical Considerations: In practice, this means that Lasso may not always be reliable for variable selection in the presence of strong multicollinearity.

Interpretation of Bayesian Regularized Regression

Lasso: Equivalent to a **Laplace Prior** on the coefficients.

Prior:

$$p(\beta_j) \propto \exp(-\lambda|\beta_j|)$$

Ridge Regression: Equivalent to a **Gaussian Prior** on the coefficients.

Prior:

$$p(\beta_j) \propto \exp(-\lambda\beta_j^2)$$

Elastic Net: Combination of Laplace and Gaussian priors.

Decision trees

Decision Trees: A tree-like model used for classification and regression.

Bagging (Bootstrap Aggregating): An ensemble method that improves the stability and accuracy of machine learning algorithms.

Boosting: An ensemble technique that combines weak learners to create a strong learner.

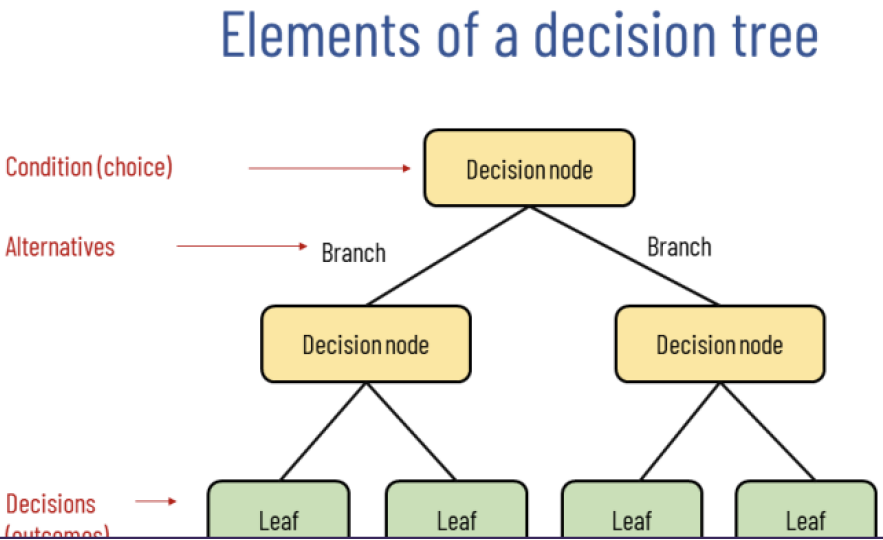
Random Forests: An ensemble of decision trees, typically trained with the bagging method.

Decision Trees Definition

A decision tree is a flowchart-like structure where an internal node represents a feature (or attribute), the branch represents a decision rule, and each leaf node represents the outcome.

Mathematical Notation

Given a dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$, where x_i is the feature vector and y_i is the target variable, a decision tree recursively splits the dataset into subsets based on feature values.


Elements of a decision tree
Condition (choice)
Decision node
Alternatives
Branch
Branch
Decision node
Decision node
Decisions
Leaf
Leaf
Leaf
Leaf

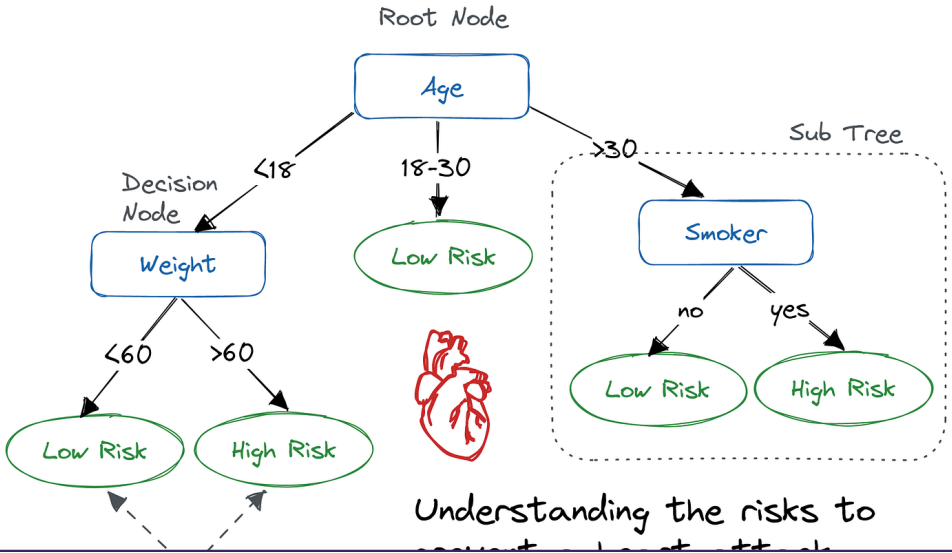
Decision trees Algorithm

Select the best feature to split the data using a criterion like Gini impurity or information gain.
Split the data into subsets.
Repeat the process recursively for each subset until a stopping condition is met (e.g., maximum depth or minimum samples per leaf).

Computational Details

Time Complexity: Building a decision tree has a time complexity of $O(n \log n)$ for balanced trees, where n is the number of samples.
Space Complexity: The space complexity is $O(n \cdot m)$, where m is the number of features.

Decision trees



Bagging Definition

Bagging, or Bootstrap Aggregating, is an ensemble method that reduces variance by averaging multiple versions of a predictor.

Mathematical Notation

Given a training set D, bagging generates B new training sets D_b by sampling with replacement. Each model f_b is trained on D_b, and the final prediction is the average (for regression) or majority vote (for classification) of all models.

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} f_b(x)$$

Bagging Algorithm

Generate B bootstrap samples from the original dataset.
Train a model on each bootstrap sample.
Aggregate the predictions from all models.

Computational Details

Time Complexity: Training B models each on n samples has a time complexity of $O(B \cdot n \log n)$.
Space Complexity: The space complexity is $O(B \cdot n \cdot m)$.

Boosting Definition

Boosting is an ensemble technique that combines weak learners to form a strong learner by focusing on the errors of previous models.

Mathematical Notation

Given a sequence of weak learners $\{f_m(x)\}_{m=1}^{M}$, boosting adjusts the weights of incorrectly classified instances and combines the weak learners into a final model:

$$F(x) = \sum_{m=1}^{M} \alpha_m f_m(x)$$

where α_m is the weight assigned to the m-th weak learner.

Boosting Algorithm

Initialize weights for all instances.
Train a weak learner on the weighted dataset.
Update the weights based on the learner's performance.
Repeat steps 2-3 for M iterations.
Combine the weak learners into a final model.

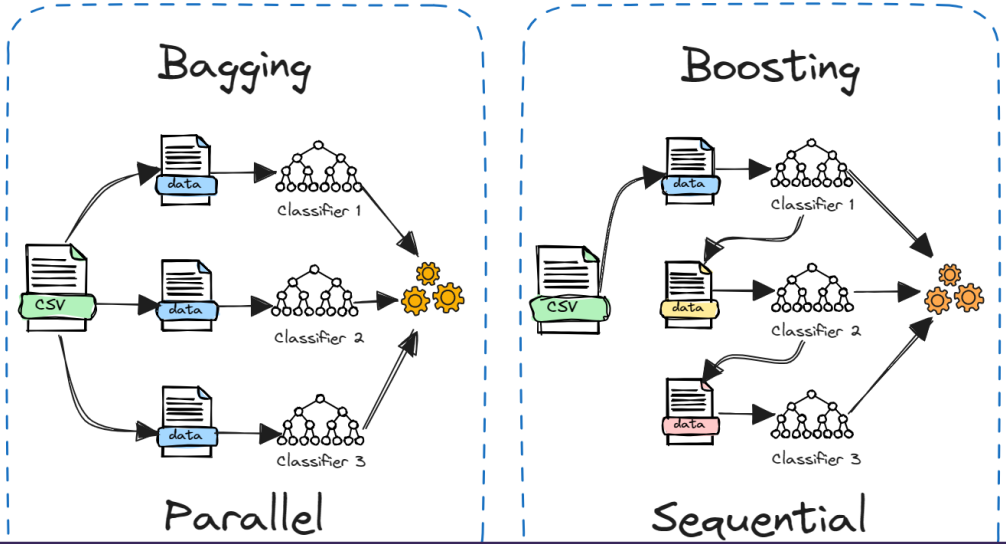
Computational Details

Time Complexity: Training M weak learners sequentially has a time complexity of $O(M \cdot n \log n)$.
Space Complexity: The space complexity is $O(M \cdot n \cdot m)$.

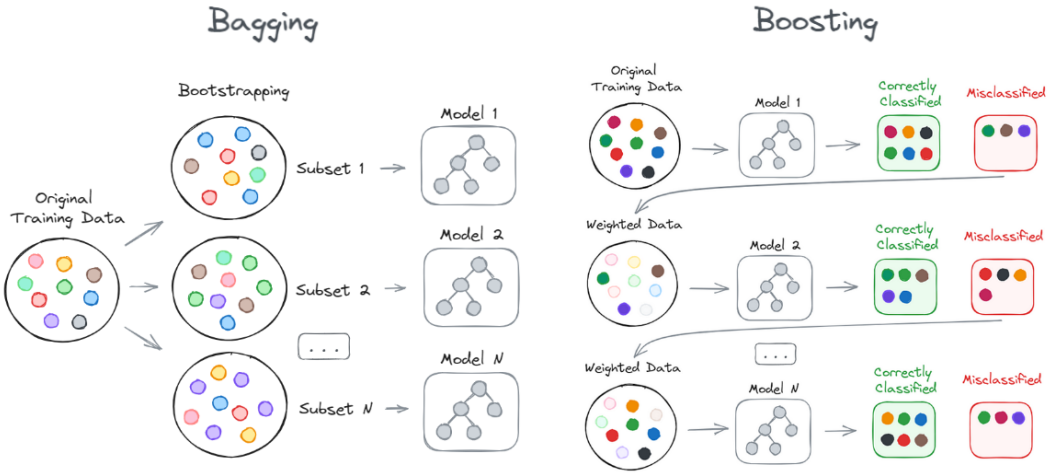
Use Cases

Improving the performance of weak models
Handling complex datasets with high bias

Bagging and boosting



Bagging and boosting



Definition

Random forests are an ensemble of decision trees, typically trained with the bagging method, where each tree is built on a random subset of features.

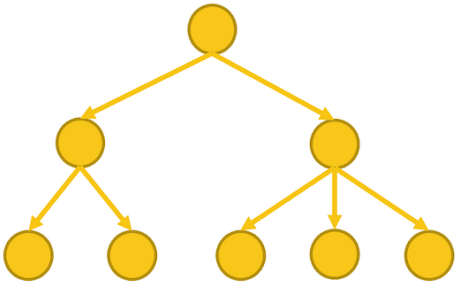
Mathematical Notation

Given B decision trees $\{T_b\}_{b=1}^{B}$, each trained on a bootstrap sample and a random subset of features, the final prediction is the average (for regression) or majority vote (for classification) of all trees:

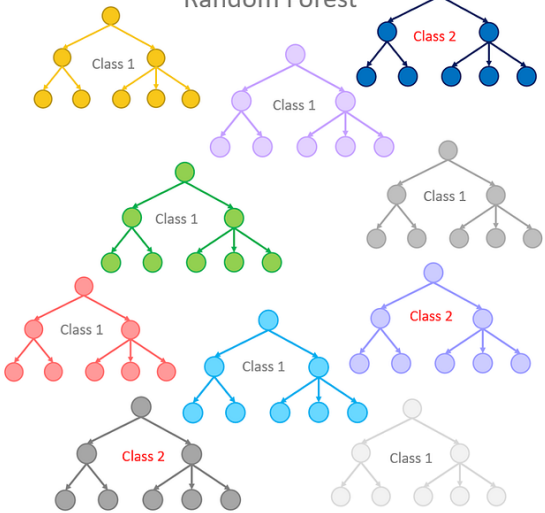
$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$$

Random Forests

Single Decision Tree



Random Forest



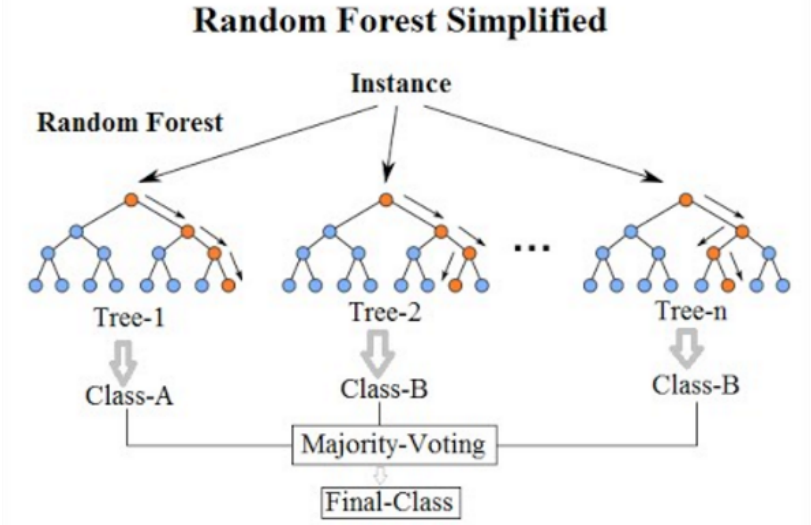
Algorithm

Generate B **bootstrap samples** from the original dataset.
For each sample, train a decision tree on a random subset of features.
Aggregate the predictions from all trees.

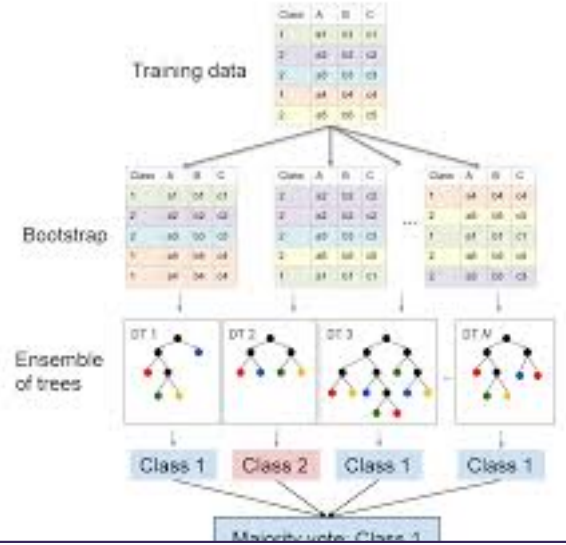
Computational Details

Time Complexity: Training B decision trees each on n samples and k features has a time complexity of $O(B \cdot n \cdot k \log k)$.
Space Complexity: The space complexity is $O(B \cdot n \cdot k)$.

Random Forests



Random Forests



Challenges with Imbalanced Datasets

Imbalanced datasets, where one class is significantly underrepresented, can lead to biased models that perform poorly on the minority class.

Handling Imbalanced Datasets in Ensemble Learning

Techniques

Resampling Methods:

Oversampling: Increase the number of instances in the minority class by duplicating them.

Undersampling: Reduce the number of instances in the majority class.

Synthetic Data Generation:

SMOTE (Synthetic Minority Over-sampling Technique): Generate synthetic samples for the minority class by interpolating between existing samples.

Cost-Sensitive Learning:

Assign higher misclassification costs to the minority class to penalize incorrect predictions more heavily.

Ensemble Methods:

Balanced Random Forests: Modify the random forest algorithm to balance the class distribution in each bootstrap sample.

EasyEnsemble and BalanceCascade: Use ensemble techniques specifically

Handling Imbalanced Datasets in Ensemble Learning

Computational Details

Time Complexity: Resampling and synthetic data generation add to the preprocessing time but do not significantly affect the training time of the ensemble methods.
Space Complexity: Oversampling increases the dataset size, which can impact memory usage.

Strengths and Weaknesses of Ensemble Methods

Strengths

Improved Accuracy: Combining multiple models often leads to better performance than individual models.
Reduced Overfitting: Ensemble methods like bagging and random forests reduce overfitting by averaging out the errors.
Robustness: Ensembles are less sensitive to the peculiarities of a single training set.

Weaknesses

Increased Complexity: Ensembles are more complex and harder to interpret than single models.
Computational Cost: Training multiple models requires more computational resources and time.
Risk of Overfitting: Boosting can lead to overfitting if not properly regularized.