Modeling Political Risk Data

COMPSS 224B: Quantitative Political Risk

Iris Malone, PhD and Mark Rosenberg, PhD

May 2, 2025

Announcements

- Problem Set 2 Due May 7
- Final Project Due May 16

Recap

Where We've Been:

- Text is context-dependent, but computational analysis is not
- Dictionary methods count frequency of words and assign weighted values
- Bag of word approaches are slowly being replaced by methods that organize text into context (themes)
- Embeddings work because neural networks create a hidden layer (word vectors) that imbue meaning

Recap

Where We've Been:

- Text is context-dependent, but computational analysis is not
- Dictionary methods count frequency of words and assign weighted values
- Bag of word approaches are slowly being replaced by methods that organize text into context (themes)
- Embeddings work because neural networks create a hidden layer (word vectors) that imbue meaning

New Terminology:

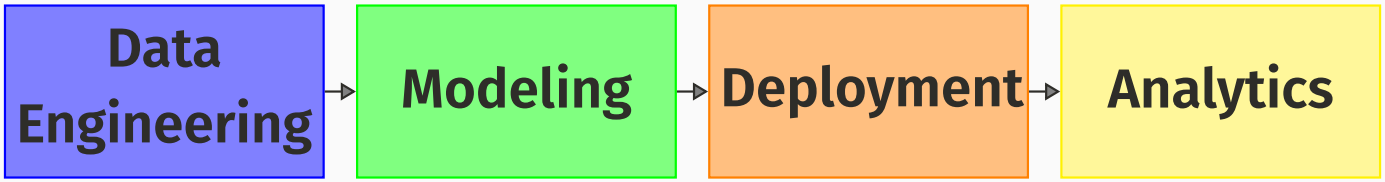
- Bag of Words
- TF-IDF
- Embeddings
- Latent Dirichlet Algorithm

Agenda

1. Common Model Challenges
2. Why We Need Non-Parametric Models
3. Random Forests
4. Boosting
5. BART
6. Special Topic: Class Imbalance

Common Model Challenges

Recall: Data Science Pipeline



Political Risk Use Cases

Unsupervised ML Products

Supervised ML Products

Political Risk Use Cases

Unsupervised ML Products

- Tools: PCA, Clustering, Text Analysis
- Risk Monitoring
- Index Creation
- Clusters
- Scorecards and Dashboards

Supervised ML Products

Political Risk Use Cases

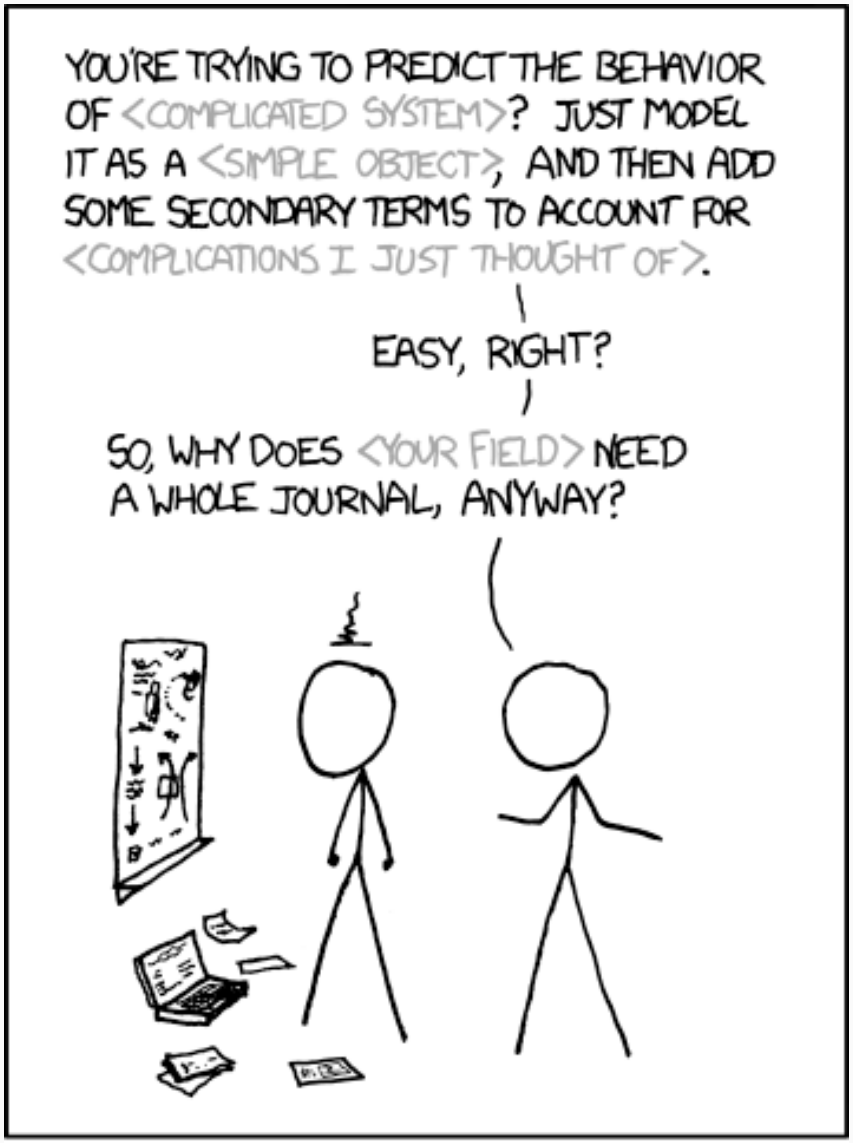
Unsupervised ML Products

- Tools: PCA, Clustering, Text Analysis
- Risk Monitoring
- Index Creation
- Clusters
- Scorecards and Dashboards

Supervised ML Products

- Tools: Models, Simulations
- Risk Management
- (Trading) Algorithms and Ideas
- Politically Oriented Risk Signals
- Forecasts
- Stress-Tests

Modelling is fun, but a black hole



7 Deadly Sins of Quantitative Political Analysis

Schrodt (2010):

1. **Pride:**
2. **Envy:**
3. **Greed:**
4. **Wrath:**
5. **Sloth:**
6. **Gluttony:**
7. **Lust:**

7 Deadly Sins of Quantitative Political Analysis

Schrodt (2010):

1. **Pride:** Overconfidence in results, no external validation
2. **Envy:**
3. **Greed:**
4. **Wrath:**
5. **Sloth:**
6. **Gluttony:**
7. **Lust:**

7 Deadly Sins of Quantitative Political Analysis

Schrodt (2010):

1. **Pride:**
2. **Envy:** Model worship and star-gazing
3. **Greed:**
4. **Wrath:**
5. **Sloth:**
6. **Gluttony:**
7. **Lust:**

7 Deadly Sins of Quantitative Political Analysis

Schrodt (2010):

1. **Pride:**
2. **Envy:**
3. **Greed:** Data snooping and massaging results
4. **Wrath:**
5. **Sloth:**
6. **Gluttony:**
7. **Lust:**

7 Deadly Sins of Quantitative Political Analysis

Schrodt (2010):

1. **Pride:**
2. **Envy:**
3. **Greed:**
4. **Wrath:** Rigid assumptions and no iteration/monitoring
5. **Sloth:**
6. **Gluttony:**
7. **Lust:**

7 Deadly Sins of Quantitative Political Analysis

Schrodt (2010):

1. **Pride:**
2. **Envy:**
3. **Greed:**
4. **Wrath:**
5. **Sloth:** Omitting variables and no hyperparameter tuning
6. **Gluttony:**
7. **Lust:**

7 Deadly Sins of Quantitative Political Analysis

Schrodt (2010):

1. **Pride:**
2. **Envy:**
3. **Greed:**
4. **Wrath:**
5. **Sloth:**
6. **Gluttony:** Kitchen-sink regressions and overfitting
7. **Lust:**

7 Deadly Sins of Quantitative Political Analysis

Schrodt (2010):

1. **Pride:**
2. **Envy:**
3. **Greed:**
4. **Wrath:**
5. **Sloth:**
6. **Gluttony:**
7. **Lust:** Confirmation or look-ahead bias

Recall: Supervised learning algorithms fall into 2 classes

1. Parametric

Recall: Supervised learning algorithms fall into 2 classes

1. Parametric
   - 1.1 More rigid $\rightarrow$ low variance
   - 1.2 Assumes f has fixed form with fixed number of parameters $(\beta_1, \ldots \beta_p)$
   - 1.3 Estimating f $\rightarrow$ estimating parameters
   - 1.4 Ex. Linear Regression Model

$$\hat{f}(X) = X_1\beta_1 + \cdots + X_p\beta_p \tag{1}$$

   - 1.5 $\hat{f}$ is less of a "black box"

Common Parametric Models

- Linear Regression
- Logistic Regression
- Probit Regression
- Linear Discriminant Analysis (LDA)
- Lasso Regression (L1 Regularization)
- Ridge Regression (L2 Regularization)
- Neural Networks

Linear Regression Model Challenges

Parametric models (e.g. linear regression) make assumptions about underlying DGP ...**that often fail.**

Linear Regression Model Challenges

Parametric models (e.g. linear regression) make assumptions about underlying DGP ...**that often fail.**

Gauss-Markov assumptions frequently violated:

1. Non-Normal Errors
2. Non-Linear Relationships
3. Variables Interact
4. Heteroskedasticity
5. Collinearity

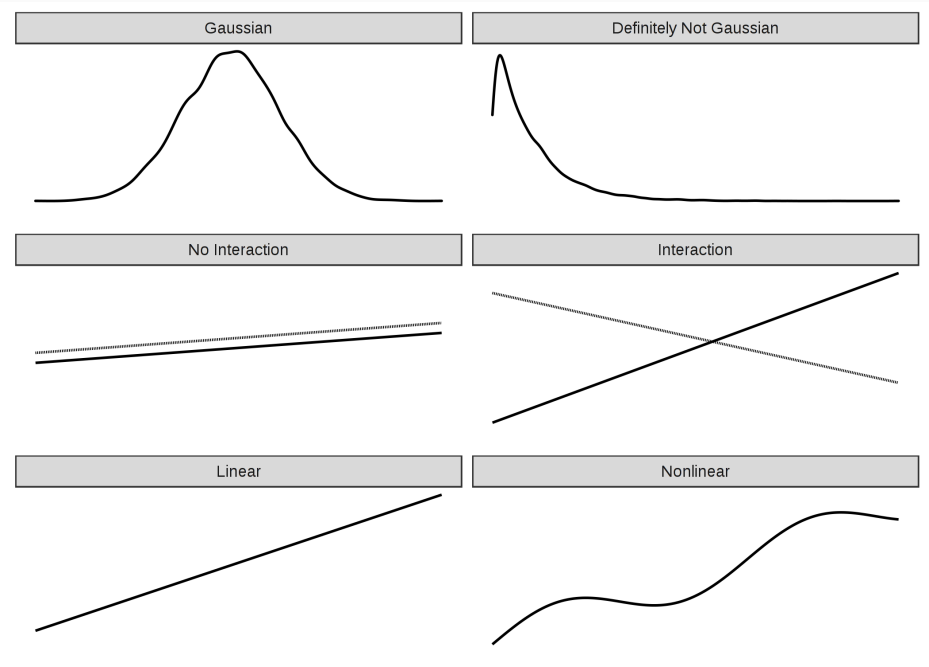


Figure 2: Christoph Molnar

FYI: This is the #1 question I got in every technical DS interview.

Logistic/Probit Regression Model Challenges

- Wrong link function $\rightarrow$ incorrect functional form (and curve-fitting method)

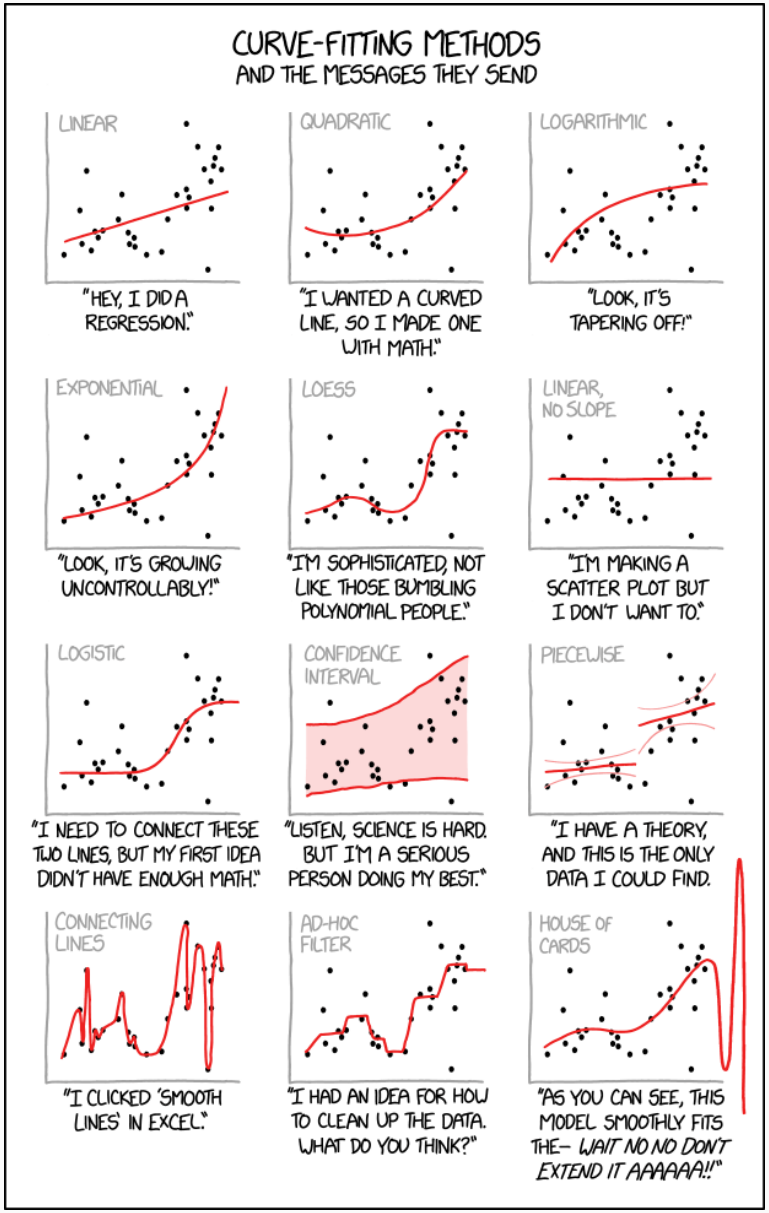


Figure 3: Source: xkcd

Logistic/Probit Regression Model Challenges

- Wrong link function $\rightarrow$ incorrect functional form (and curve-fitting method)
- Multimodal data $\neq$ single probability distribution

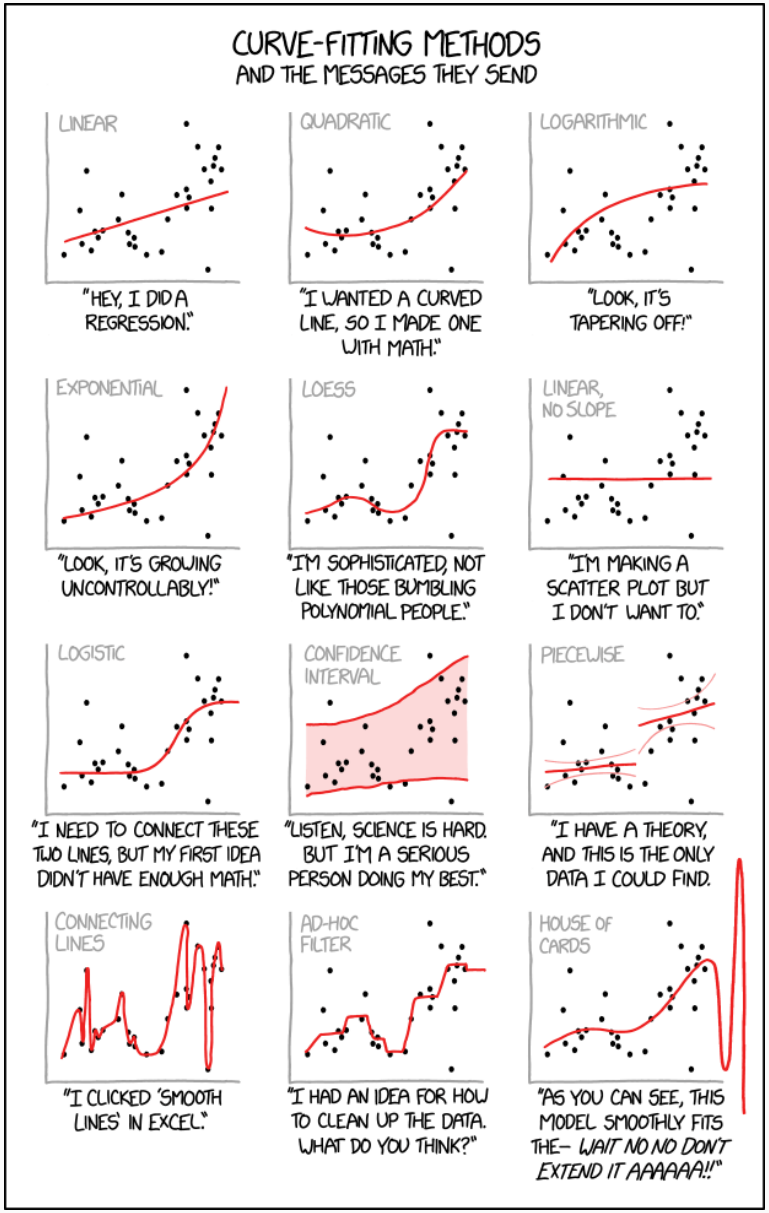


Figure 3: Source: xkcd

Logistic/Probit Regression Model Challenges

- Wrong link function $\rightarrow$ incorrect functional form (and curve-fitting method)
- Multimodal data $\neq$ single probability distribution
- Well-separated class $\rightarrow$ unstable coefficients

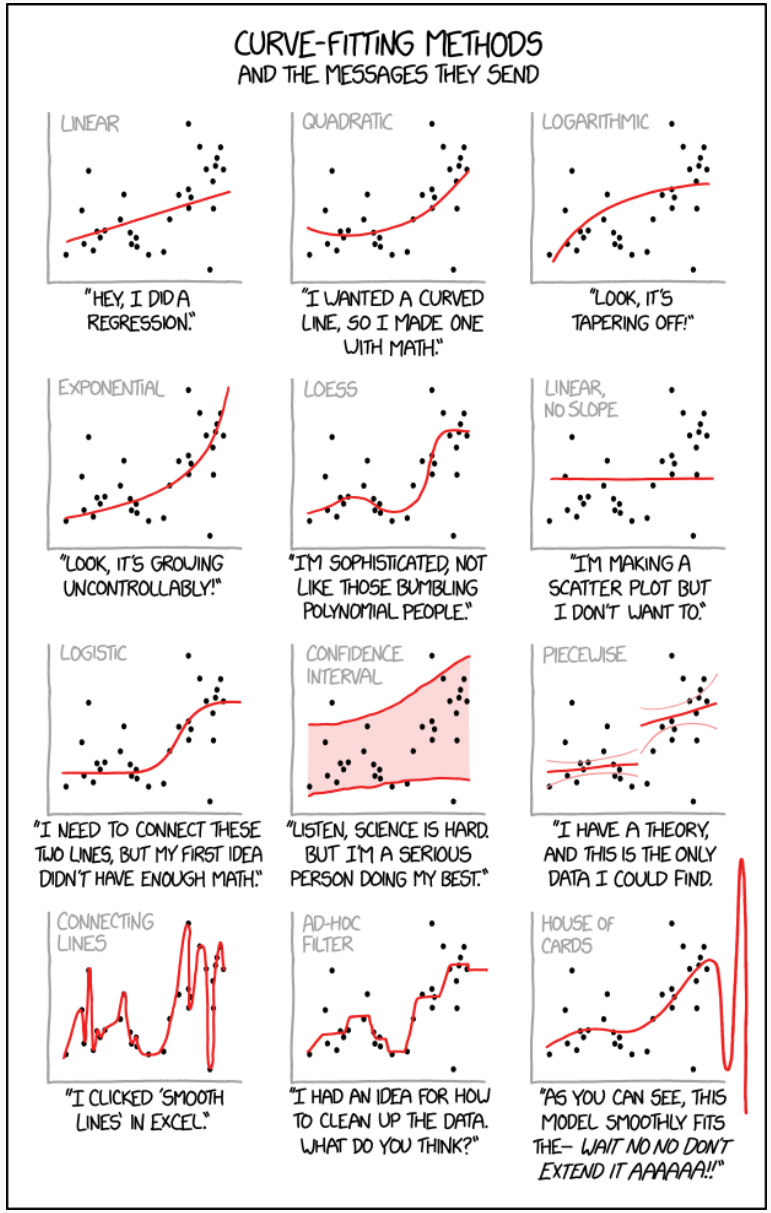


Figure 3: Source: xkcd

Logistic/Probit Regression Model Challenges

- Wrong link function $\rightarrow$ incorrect functional form (and curve-fitting method)
- Multimodal data $\neq$ single probability distribution
- Well-separated class $\rightarrow$ unstable coefficients
- Imbalanced classes $\rightarrow$ biased predictions toward the majority class

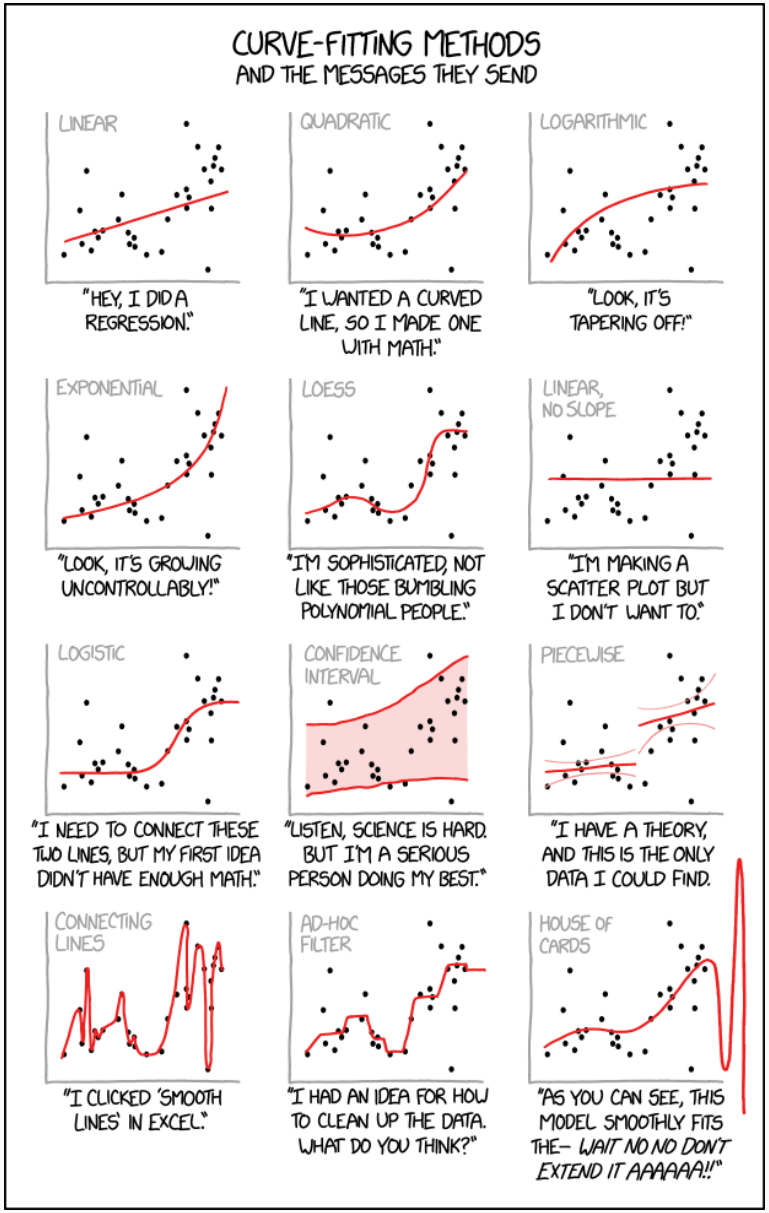


Figure 3: Source: xkcd

3 Solutions to Common Regression Problems

- Variables often interact

3 Solutions to Common Regression Problems

- Variables often interact $\rightarrow$ **Solution:** Add an interaction parameter

3 Solutions to Common Regression Problems

- Variables often interact $\rightarrow$ **Solution:** Add an interaction parameter
- Residuals not perfectly bell-shaped

3 Solutions to Common Regression Problems

- Variables often interact $\rightarrow$ **Solution:** Add an interaction parameter
- Residuals not perfectly bell-shaped $\rightarrow$ **Solution:** Semi-Parametric Models, e.g. GAMs

3 Solutions to Common Regression Problems

- Variables often interact $\rightarrow$ **Solution:** Add an interaction parameter
- Residuals not perfectly bell-shaped $\rightarrow$ **Solution:** Semi-Parametric Models, e.g. GAMs
- Relationships are not strictly linear

3 Solutions to Common Regression Problems

- Variables often interact $\rightarrow$ **Solution:** Add an interaction parameter
- Residuals not perfectly bell-shaped $\rightarrow$ **Solution:** Semi-Parametric Models, e.g. GAMs
- Relationships are not strictly linear $\rightarrow$ **Solution:** Non-Parametric Models

Why We Need Non-Parametric Models

Motivation: Bias-Variance Trade-Off

- A central ML challenge is finding a method that minimizes *both* variance and bias.
- Bias-Variance Trade-Off: Models tend to result in either (1) low variance and high bias (under-fitting) or (2) high variance and low bias (over-fitting).
- Parametric and non-parametric methods take different approaches to optimize bias-variance trade-off

Supervised learning algorithms fall into 2 classes

1. Parametric
   - 1.1 More rigid $\rightarrow$ low variance
   - 1.2 Assumes f has fixed form with fixed number of parameters $(\beta_1, \ldots \beta_p)$
   - 1.3 Estimating f $\rightarrow$ estimating parameters
   - 1.4 Ex. Linear Regression Model

$$\hat{f}(X) = X_1\beta_1 + \cdots + X_p\beta_p \tag{2}$$

   - 1.5 $\hat{f}$ is less of a "black box"
2. Non-Parametric

Supervised learning algorithms fall into 2 classes

1. Parametric
   - 1.1 More rigid $\rightarrow$ low variance
   - 1.2 Assumes f has fixed form with fixed number of parameters $(\beta_1, \ldots \beta_p)$
   - 1.3 Estimating f $\rightarrow$ estimating parameters
   - 1.4 Ex. Linear Regression Model

$$\hat{f}(X) = X_1\beta_1 + \cdots + X_p\beta_p \tag{2}$$

   - 1.5 $\hat{f}$ is less of a "black box"
2. Non-Parametric
   - 2.1 More flexible $\rightarrow$ low bias
   - 2.2 No fixed f to describe data
   - 2.3 $\hat{f}$ is "black box"

Common Non-Parametric Models

- K-Nearest Neighbors
- Classification and Regression Trees (CART)
  - Bagging
  - Random Forests
  - Boosting
  - Bayesian Additive Regression Trees
- Support Vector Machines

Why Use Non-Parametric Models?

1. **Avoid Common Regression Problems**
2. Explore New/Original Data
3. Increased Transparency

Why Use Non-Parametric Models?

1. Avoid Common Regression Problems
2. **Explore New/Original Data**
3. **Increased Transparency**

Explore New/Original Data Collection

- Novelty ($n = 1$) Data Problems:
  - What's the underlying DGP?
  - What are the most relevant predictors?
  - What's the true f?

Explore New/Original Data Collection

- Novelty ($n = 1$) Data Problems:
  - What's the underlying DGP?
  - What are the most relevant predictors?
  - What's the true f?
- Potential Solutions:
  - Model everything

Explore New/Original Data Collection

- Novelty ($n = 1$) Data Problems:
  - What's the underlying DGP?
  - What are the most relevant predictors?
  - What's the true f?
- Potential Solutions:
  - Model everything but high risk of overfitting

Explore New/Original Data Collection

- Novelty ($n = 1$) Data Problems:
  - What's the underlying DGP?
  - What are the most relevant predictors?
  - What's the true f?
- Potential Solutions:
  - Model everything but high risk of overfitting
  - Use existing lit to make predictions for variable inputs

Explore New/Original Data Collection

- Novelty ($n = 1$) Data Problems:
  - What's the underlying DGP?
  - What are the most relevant predictors?
  - What's the true f?
- Potential Solutions:
  - Model everything but high risk of overfitting
  - Use existing lit to make predictions for variable inputs but could make wrong (biased?) assumptions

Explore New/Original Data Collection

- Novelty ($n = 1$) Data Problems:
  - What's the underlying DGP?
  - What are the most relevant predictors?
  - What's the true f?
- Potential Solutions:
  - Model everything but high risk of overfitting
  - Use existing lit to make predictions for variable inputs but could make wrong (biased?) assumptions
  - **Use non-parametric methods**

Increased Transparency

- Parametric Model Ethics
  - Data Dredging (Sin of Lust)
  - Model P-Hacking (Sin of Greed)
  - Stargazing (Sin of Envy)

Increased Transparency

- Parametric Model Ethics
  - Data Dredging (Sin of Lust)
  - Model P-Hacking (Sin of Greed)
  - Stargazing (Sin of Envy)
- Potential Solutions:
  - Model everything but high risk of overfitting
  - Use existing lit to make predictions for variable selection

Increased Transparency

- Parametric Model Ethics
  - Data Dredging (Sin of Lust)
  - Model P-Hacking (Sin of Greed)
  - Stargazing (Sin of Envy)
- Potential Solutions:
  - Model everything but high risk of overfitting
  - Use existing lit to make predictions for variable selection but could make wrong (biased?) assumptions

Increased Transparency

- Parametric Model Ethics
  - Data Dredging (Sin of Lust)
  - Model P-Hacking (Sin of Greed)
  - Stargazing (Sin of Envy)
- Potential Solutions:
  - Model everything but high risk of overfitting
  - Use existing lit to make predictions for variable selection but could make wrong (biased?) assumptions
  - **Use non-parametric methods**

Non-Parametric Model Challenges

- **Lack of Interpretability:** Black box decision-making $\rightarrow$ risk post hoc rationalization

Figure 4: Source: Pablo Picasso. Nice, but can't bring to the C-Suite.

Non-Parametric Model Challenges

- **Lack of Interpretability:** Black box decision-making $\rightarrow$ risk post hoc rationalization
- **Overfitting:** Highly susceptible due to being overly flexible.

Figure 4: Source: Pablo Picasso. Nice, but can't bring to the C-Suite.

Non-Parametric Model Challenges

- **Lack of Interpretability:** Black box decision-making $\rightarrow$ risk post hoc rationalization
- **Overfitting:** Highly susceptible due to being overly flexible.
- **Hyperparameter Tuning:** Many models depend on choosing an 'optimal' (subjective) specification (e.g. k-neighbors, d-degrees flexibility, tuning parameter)

Figure 4: Source: Pablo Picasso. Nice, but can't bring to the C-Suite.

3 Solutions to Non-Parametric Model Challenges

- Overfitting

3 Solutions to Non-Parametric Model Challenges

- Overfitting $\rightarrow$ **Solution:** Cross-validation

3 Solutions to Non-Parametric Model Challenges

- Overfitting $\rightarrow$ **Solution:** Cross-validation
- Hyperparameter Tuning

3 Solutions to Non-Parametric Model Challenges

- Overfitting $\rightarrow$ **Solution:** Cross-validation
- Hyperparameter Tuning $\rightarrow$ **Solution:** grid search

3 Solutions to Non-Parametric Model Challenges

- Overfitting $\rightarrow$ **Solution:** Cross-validation
- Hyperparameter Tuning $\rightarrow$ **Solution:** grid search
- Lack Interpretability

3 Solutions to Non-Parametric Model Challenges

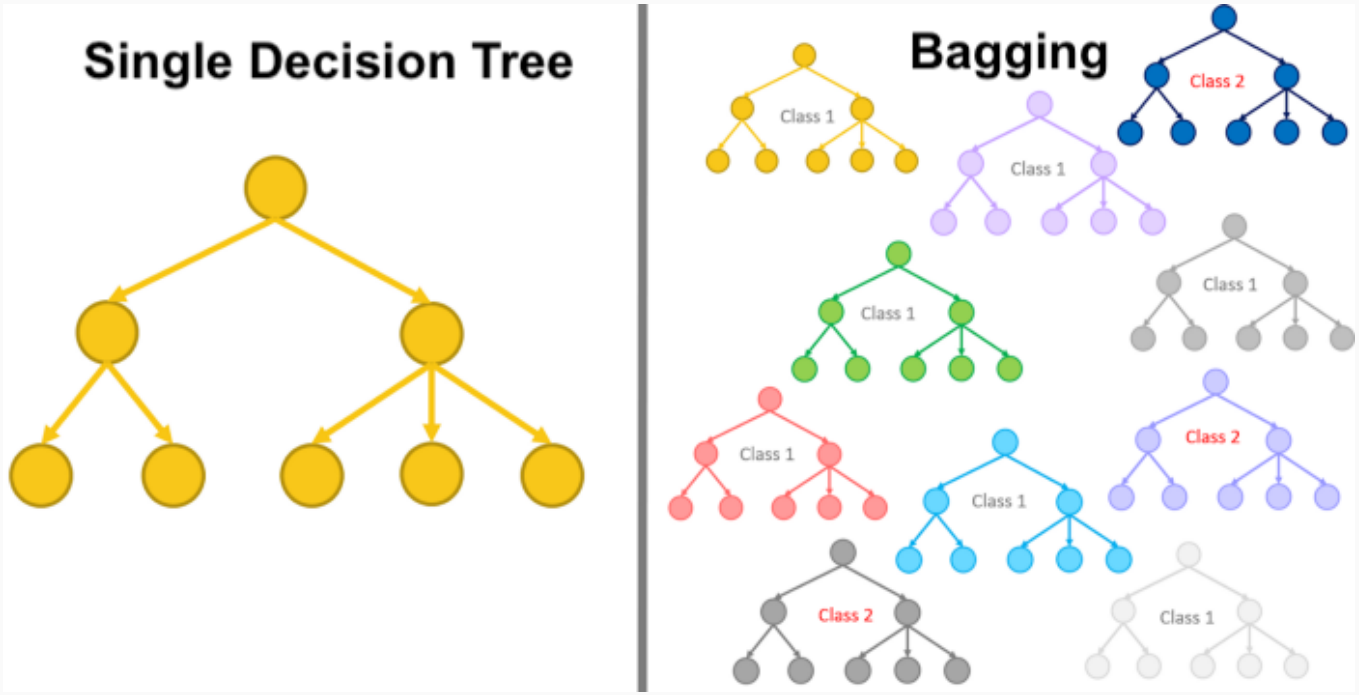
- Overfitting $\rightarrow$ **Solution:** Cross-validation
- Hyperparameter Tuning $\rightarrow$ **Solution:** grid search
- Lack Interpretability $\rightarrow$ **Solution:** Interpretable ML (next week)

Random Forests

Recap: Bagging popular alternative to (single) decision tree

Bagging = Bootstrap Aggregating

Main Idea: This is an **ensemble method** in which we create lots of different decision trees using bootstrap and average the predictions together. It reduces high variance problem of single tree by averaging lots of decision trees together



Problems with Bagging

Main Limit to Bagging: Trees produced by bootstrap look very similar. Why?

Problems with Bagging

Main Limit to Bagging: Trees produced by bootstrap look very similar. Why?

- Bootstrapped samples $\rightarrow$ each tree built around different observations i, but ...

Problems with Bagging

Main Limit to Bagging: Trees produced by bootstrap look very similar. Why?

- Bootstrapped samples $\rightarrow$ each tree built around different observations i, but ...
- Model examines same number/type of predictors X_j

Problems with Bagging

Main Limit to Bagging: Trees produced by bootstrap look very similar. Why?

- Bootstrapped samples $\rightarrow$ each tree built around different observations i, but ...
- Model examines same number/type of predictors X_j
- This results in slight difference across decision rules, but generally similar decision rules

Problems with Bagging

Main Limit to Bagging: Trees produced by bootstrap look very similar. Why?

- Bootstrapped samples $\rightarrow$ each tree built around different observations i, but ...
- Model examines same number/type of predictors X_j
- This results in slight difference across decision rules, but generally similar decision rules
- Similar decision rules $\rightarrow$ **correlated trees**

Limits to Correlated Trees

Problem: Correlated trees produce potentially biased results ...

- If there is one highly influential predictor $\rightarrow$ tendency to prune all other predictors
- If there are collinear predictors $\rightarrow$ bias to variance-maximizing predictor
- If there is combination of continuous and binary measures $\rightarrow$ bias to continuous predictors

Limits to Correlated Trees

Problem: Correlated trees produce potentially biased results ...

- If there is one highly influential predictor $\rightarrow$ tendency to prune all other predictors
- If there are collinear predictors $\rightarrow$ bias to variance-maximizing predictor
- If there is combination of continuous and binary measures $\rightarrow$ bias to continuous predictors

Solution: **Random Forests**

Random Forests

Main Idea: Create a large number of bootstrapped decision trees, but vary the number of predictors you feed each tree in order to **decorrelate** the predictions.

Random Forests

Main Idea: Create a large number of bootstrapped decision trees, but vary the number of predictors you feed each tree in order to **decorrelate** the predictions.

Loss Function:

- Regression Problem: RSS
- Classification Problem: $0-1$ Loss, Gini Index, Cross-Entropy

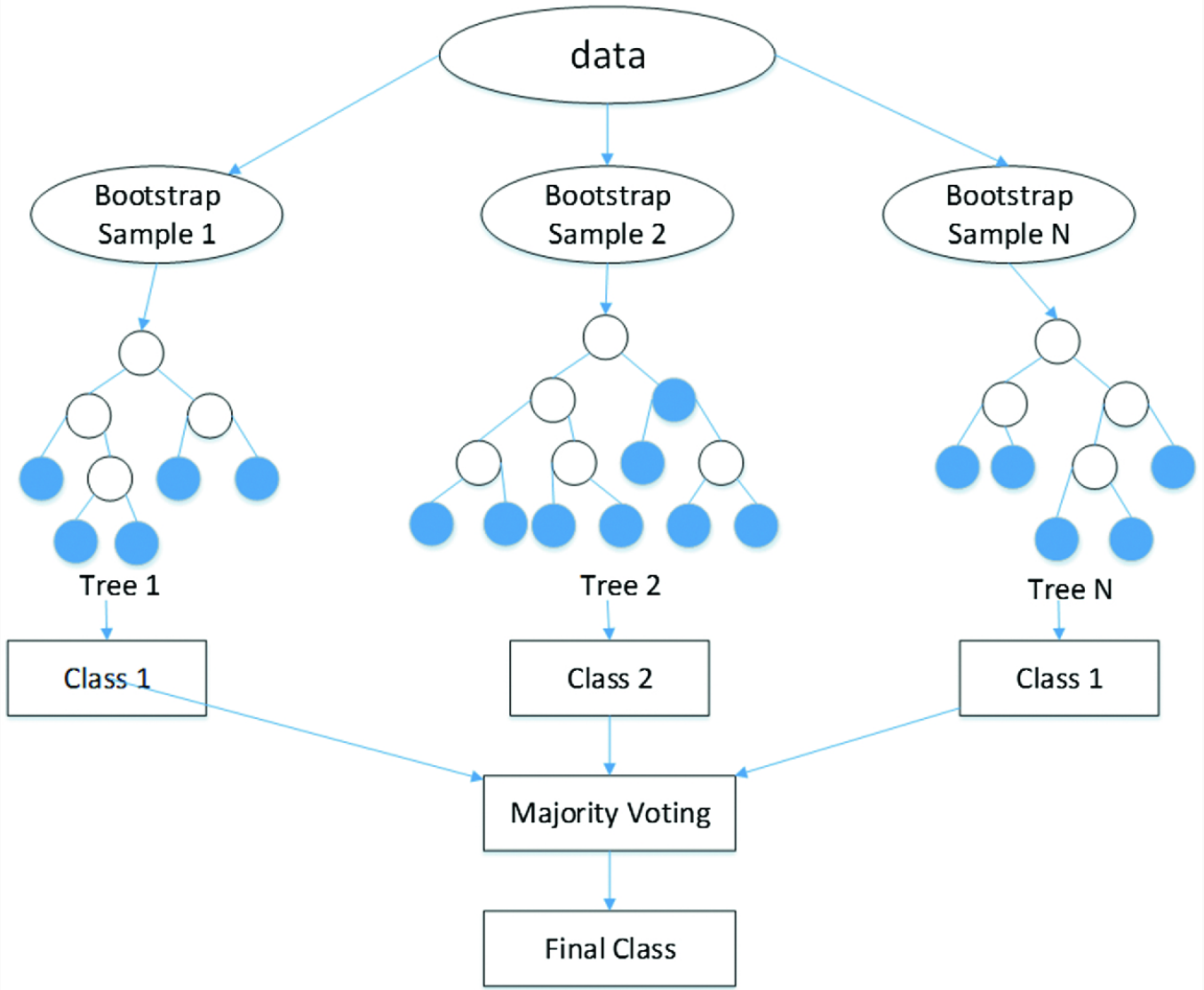
Random Forests

Procedure:

- Create different bootstrap samples B to build a decision tree
- When growing the tree, select a random sample of $m \in [1, p]$ predictors to consider in each step
- Build the tree to minimize preferred loss function
- Average the prediction of each tree $\rightarrow$ majority class votes

Random Forests

Selecting a random sample of $m \in [1, p]$ predictors to consider in each step leads to very different ("uncorrelated") trees each time.



Predictions with Random Forest

Regressors make predictions by **averaging predictions** of each tree

Predictions with Random Forest

Regressors make predictions by **averaging predictions** of each tree
Classifiers make predictions by **majority class** voting:

Predictions with Random Forest

Regressors make predictions by **averaging predictions** of each tree
Classifiers make predictions by **majority class** voting:

Predictions with Random Forest

Regressors make predictions by **averaging predictions** of each tree
Classifiers make predictions by **majority class** voting:

Example: Build decision tree with variables age, highest education, favorite baseball team to predict whether unknown participant is student vs professor

- Decision Tree 1 uses only feature Age (Age):
  - If Age < 30, predict student, otherwise predict professor
- Decision Tree 2 uses only feature Edu (Education):
  If Education $=$ BA Degree, predict student, otherwise predict professor
- Decision Tree 3 uses only feature $Sports$ (Baseball Team):
  If Sports $=$ Dodgers, predict student, otherwise predict professor

Predictions with Random Forest

Suppose we want to predict the class of a new point with the following features: (Age =25, Edu=BA, $Sports$=SF Giants). Get predictions from each separate decision tree and average:

- Decision Tree 1 sees $Age = 25$ and predicts Class=student
- Decision Tree 2 sees $Edu =$ BA Degree and predicts Class=student
- Decision Tree 3 sees $Sports =$ SF Giants and predicts Class=professor

There were 2 votes for student and 1 vote for professor, so the forest predicts the unknown observer is student, the class that received the majority of the votes.

Hyperparameter Tuning

Key hyper-parameters in Random Forests:

- B, or the number of distinct trees
- m, or the number of variables we put into each model

Hyperparameter Tuning

Key hyper-parameters in Random Forests:

- B, or the number of distinct trees
- m, or the number of variables we put into each model

How to choose the optimal B?

Hyperparameter Tuning

Key hyper-parameters in Random Forests:

- B, or the number of distinct trees
- m, or the number of variables we put into each model

How to choose the optimal B?

- Rule of Thumb: $\sim 500 - 1000$ trees
- Cross-Validation

Hyperparameter Tuning

Key hyper-parameters in Random Forests:

- B, or the number of distinct trees
- m, or the number of variables we put into each model

How to choose the optimal B?

- Rule of Thumb: $\sim 500 - 1000$ trees
- Cross-Validation

How to choose the optimal m?

Hyperparameter Tuning

Key hyper-parameters in Random Forests:

- B, or the number of distinct trees
- m, or the number of variables we put into each model

How to choose the optimal B?

- Rule of Thumb: $\sim 500 - 1000$ trees
- Cross-Validation

How to choose the optimal m?

- Rule of Thumb: $m = \sqrt{p}$
- Cross-Validation

Advantages and Disadvantages to Random Forests

Advantages:

Disadvantages:

Advantages and Disadvantages to Random Forests

Advantages:

- Very popular (“leatherman of learning”)
- Very customizable and easy to tune
- Performs better than bagging and CART
- Lots of tools for model evaluation and assessment (separation plots, ePCP, Brier Scores)

Disadvantages:

Advantages and Disadvantages to Random Forests

Advantages:

- Very popular ("leatherman of learning")
- Very customizable and easy to tune
- Performs better than bagging and CART
- Lots of tools for model evaluation and assessment (separation plots, ePCP, Brier Scores)

Disadvantages:

- Large trees $\rightarrow$ slow and computationally expensive
- Does not perform as well as other algorithms
- Top-down recursive splitting $\rightarrow$ suboptimal branches
- Assumes independence between observations $\rightarrow$ no learning
- Can't handle time-dependencies or sequences of data!

Boosting

Gradient Boosting Methods (GBM)

Main Idea: Grow trees sequentially to **learn** from results of previous trees

- First use the samples that are easiest to predict and make splits
- Learn trends in the remaining data to update splits and "boost" performance
- Iteratively move on to harder samples until can no longer minimize tree error

How does the model learn?

Model slowly learns by examining the residuals rather than the outcome when making decision rule splits

Procedure (in words):

- Input all parameters into the model and estimate base model
- Fit a decision tree which tries to predict residuals ($y - \hat{y}$) from base model
- Add this decision tree to the fitted function $\hat{f}$ and update the new estimated residuals
- Iteratively fit trees to the (increasingly smaller) residuals in order to improve $\hat{f}$

GBM Loss Function

Uses idea of **gradient descent algorithm** to minimize error by fitting ensemble of trees via gradient loss

Procedure (in math):

- Set $\hat{f}(x) = 0$ and $r_i = y_i$ for $i = 1, \ldots, n$
- For each tree $b = 1, \ldots, B$ iterate:
  - Fit a decision tree $\hat{f}^b$ with d splits to the response $r_1, \ldots, r_n$
  - Update the prediction to:

$$\hat{f} + \lambda \hat{f}^b \to \hat{f}$$

  - Update the residuals:

$$r_i + \lambda \hat{f}^b \to r_i$$

- Iterate until residuals no longer minimized
- Output the final model:

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

Hyperparameters

Boosting has 3 tuning parameters:

1. **Number of Trees (B):**

2. **Learning Rate (λ):**

3. **Interaction Depth (d)**

Hyperparameters

Boosting has 3 tuning parameters:

1. **Number of Trees (B):**

   Smaller B sometimes better. Unlike RF, higher risk of overfitting as number of trees grow.

2. **Learning Rate (λ):**

3. **Interaction Depth (d)**

Hyperparameters

Boosting has 3 tuning parameters:

1. **Number of Trees (B):**

2. **Learning Rate (λ):**

   Shrinkage parameter controls how slowly the model learns, typically 0.01 or 0.001

3. **Interaction Depth (d)**

Hyperparameters

Boosting has 3 tuning parameters:

1. **Number of Trees (B):**

2. **Learning Rate (λ):**

3. **Interaction Depth (d)**

   Number of splits controls the complexity of the ensemble.
   Higher values of d producing more complicated (deeper) trees

GBM vs Random Forests

Random Forests

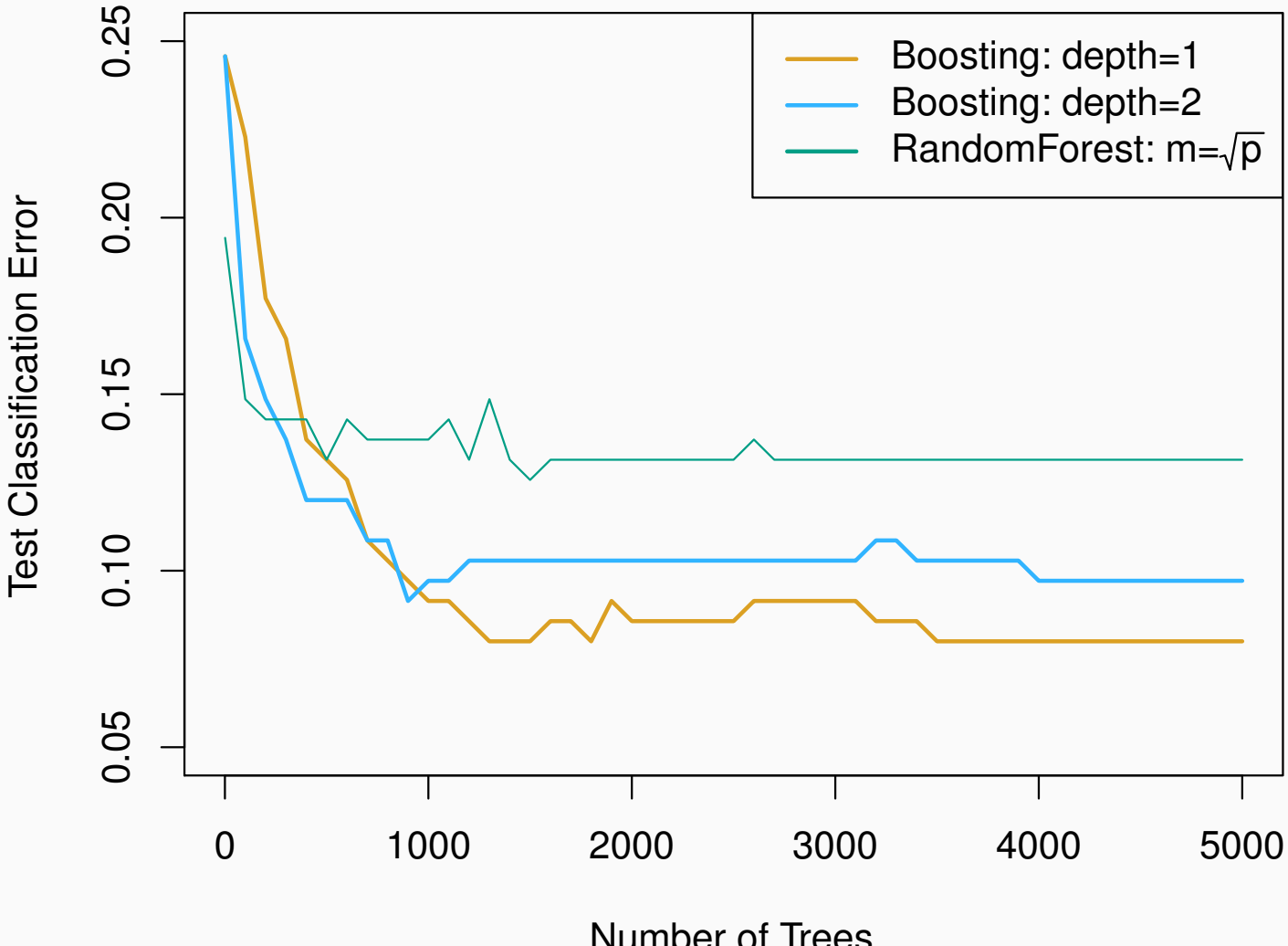
- involves bootstrap sampling $\rightarrow$ independence between trees
- no learning
- high risk of overfitting

GBM

- does not involve bootstrap sampling $\rightarrow$ dependence between trees
- slow learning
- less risk of overfitting

GBM vs Random Forests

Boosting tends to perform better than random forest



Common Decision Tree Challenges

- **Categorical Predictors:** How does the model partition categorical predictors?
- **Continuous vs Binary Predictors:** How does the model treat continuous vs binary predictors?
- **Missing Data:** How does the model handle missing data?

Categorical Predictors

- If there are only 2 categories, then the split is obvious. We don't have to choose the splitting point s for a numeric variable

Categorical Predictors

- If there are only 2 categories, then the split is obvious. We don't have to choose the splitting point s for a numeric variable
- If there are more than 2 categories ...
  - Order the categories according to the average of the response, e.g. if c most common $\in [a, b, c]$, then set level $a > c > b$
  - Transform categories into numeric variable with this ordering and choose a splitting point s

Continuous vs Binary Predictors

- If the model contains one type of predictor (e.g. all continuous or all binary), then no worry

Continuous vs Binary Predictors

- If the model contains one type of predictor (e.g. all continuous or all binary), then no worry
- If the model contains different types of predictors, then worry ...

Continuous vs Binary Predictors

- If the model contains one type of predictor (e.g. all continuous or all binary), then no worry
- If the model contains different types of predictors, then worry ...
  - Model will prioritize continuous over binary variables $\rightarrow$ easier to partition because more possible splitting points
  - Resulting tree will have continuous variables higher on tree than binary variables
  - **Risk:** False impression continuous variables more important than binary variables!

Continuous vs Binary Predictors

Correct for mixture of continuous and binary predictors using **conditional inference tree**

- Uses alternative recursive partitioning algorithm
- Uses alternative significance test instead of Gini index to find relevant predictors

Missing Data

- **Problem:** If a sample is missing variable X_j and a tree contains a split according to $X_j > s$ then we may not be able to assign the sample to a region

Missing Data

- **Problem:** If a sample is missing variable X_j and a tree contains a split according to $X_j > s$ then we may not be able to assign the sample to a region
- **Solution 1:** Create contingent decision rules
  - Propagate a sample down the tree
  - When choosing a new split with variable X_j (growing the tree)
    - only consider the samples which have the variable X_j
    - in addition to choosing the best split, choose a second best split using a different variable, and a third best, ...
  - If the observation is missing a variable, try the second best decision rule. If missing both variables, try the third best decision rule, etc.
- **Solution 2:** Imputation (depending on type of missingness)

BART

Recall: Bayes Theorem

Our predicted probability depends on available data and the model we use to fit the data.

$$\textit{outcome} \propto \textit{model} \times \textit{data}$$

- $p(x)$: prior probability (data)
- $p(x \mid y)$: likelihood function (model)
- $p(y \mid x)$: posterior probability (outcome)

$$\textit{posterior probability} \propto \textit{likelihood} \times \textit{prior probability}$$

$$p(y \mid x) \propto p(x \mid y)p(x) = \frac{p(y) \cdot p(x \mid y)}{p(x)}$$

Bayesian Additive Regression Trees (BART)

Main Idea: BART is similar to GBM in that it sums the contribution of sequential weak learners.
Procedure:

- Builds a series of simple decision trees.
- Uses an iterative backfitting algorithm to cycle over and over through the B trees in order to learn
- Model sequentially learns which variables are most important..
  - In GBM, each sequential tree is multiplied by learning rate λ
  - In BART, model uses prior beliefs (σ) to iteratively update (and guide) posterior probability predictions

Prior Beliefs in BART

- $\sigma \sim Unif$: Each variable in the classifier initially has an equal probability of inclusion as a splitting variable.
  - σ is relatively non-informative
  - Posterior predictions returns estimates based on $\hat{f}$ (similar to a conventional frequentist approach)
- $\sigma \sim Inv - \chi^2$ distribution: Initial beliefs more important than tree decision rules until model learns enough that $\hat{f}$ drives posterior more than σ

BART Performance

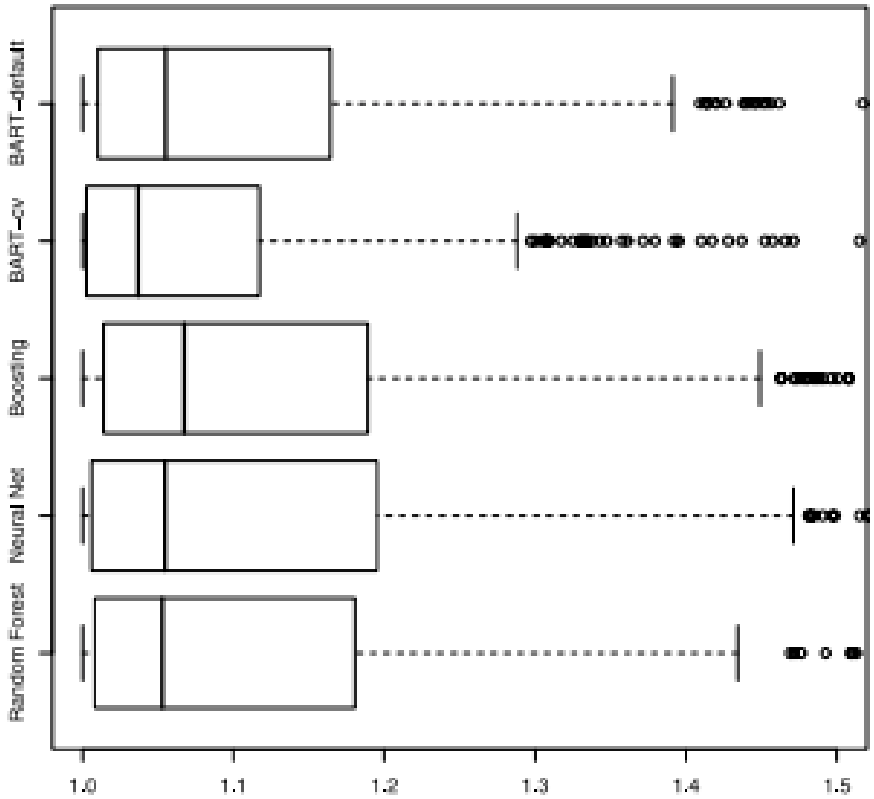


FIG. 2. *Boxplots of the RRMSE values for each method across the* 840 *test/train splits. Percentage RRMSE values larger than* 1.5 *for each method (and not plotted) were the following: random forests* 16.2%, *neural net* 9.0%, *boosting* 13.6%, *BART-cv* 9.0% *and BART-default* 11.8%. *The Lasso (not plotted because of too many large RRMSE values) had* 29.5% *greater than* 1.5.

Figure 5: Smaller error than Boosting, NN, and RF. Chipman et al. (2010), p. 281

Hyperparameters

BART has 3 tuning parameters:

1. **Number of Trees (B):**

2. **Prior Belief (σ):**
3. **Alpha (α):**

Hyperparameters

BART has 3 tuning parameters:

1. **Number of Trees (B):** Model needs large number of trees to learn and converge on model
2. **Prior Belief (σ):**
3. **Alpha (α):**

Hyperparameters

BART has 3 tuning parameters:

1. **Number of Trees (B):**

2. **Prior Belief (σ):** Generally the inverse chi-squared distribution
3. **Alpha (α):**

Hyperparameters

BART has 3 tuning parameters:

1. **Number of Trees (B):**

2. **Prior Belief (σ):**
3. **Alpha (α):** Threshold for variable inclusion $\rightarrow$ variable selection

Variable Selection

- For each iteration, the model returns a **variable inclusion proportion**, which records the number of times a variable appears in a given tree.
- Higher variable inclusion proportions indicate more important variables.
- For each variable, BART creates a distribution of inclusion proportions across a series of permutations.

Variable Inclusion Thresholds

- **Local Distribution:** Identifies a variable x as relevant if its inclusion proportion falls above the $1 - \alpha$ quantile of the permutation distribution for x.
- **Global Distribution:** Stricter approach; identifies a variable x as relevant if its inclusion proportion falls above the $1 - \alpha$ quantile of the base model distribution for x.

Advantages and Disadvantages to Tree-Based Methods

Advantages:

Disadvantages:

Advantages and Disadvantages to Tree-Based Methods

Advantages:

- Avoid problems of parametric regression methods
- Mimic human decision-making
- Easy to visualize
- Easily handle qualitative predictors and missing data

Disadvantages:

Advantages and Disadvantages to Tree-Based Methods

Advantages:

- Avoid problems of parametric regression methods
- Mimic human decision-making
- Easy to visualize
- Easily handle qualitative predictors and missing data

Disadvantages:

- Likely to overfit to noisy data
- Greedy algorithm can lead to worse fit
- High variance $\rightarrow$ different results

Special Topic: Class Imbalance

Motivation: Model performance metrics are not always meaningful

"Good" model performs better than No Information Rate (NIR)

No Information Rate (NIR):

Accuracy: Overall Classification Rate

Sensitivity/Recall: True Positive Rate

Specificity: True Negative Rate

Motivation: Model performance metrics are not always meaningful

"Good" model performs better than No Information Rate (NIR)

No Information Rate (NIR):

- Predicted accuracy if we always predicted majority class (0)
- Imbalanced data (large majority class) $\rightarrow$ large NIR

Accuracy: Overall Classification Rate

Sensitivity/Recall: True Positive Rate

Specificity: True Negative Rate

Motivation: Model performance metrics are not always meaningful

"Good" model performs better than No Information Rate (NIR)

No Information Rate (NIR):

- Predicted accuracy if we always predicted majority class (o)
- Imbalanced data (large majority class) $\rightarrow$ large NIR

Accuracy: Overall Classification Rate

- Good for general model performance
- Bad model: accuracy $<$ NIR

Sensitivity/Recall: True Positive Rate

Specificity: True Negative Rate

Motivation: Model performance metrics are not always meaningful

"Good" model performs better than No Information Rate (NIR)

No Information Rate (NIR):

- Predicted accuracy if we always predicted majority class (0)
- Imbalanced data (large majority class) $\rightarrow$ large NIR

Accuracy: Overall Classification Rate

- Good for general model performance
- Bad model: accuracy $<$ NIR

Sensitivity/Recall: True Positive Rate

- Strive for high sensitivity
- Will be **low** if there is class imbalance

Specificity: True Negative Rate

Motivation: Model performance metrics are not always meaningful

"Good" model performs better than No Information Rate (NIR)

No Information Rate (NIR):

- Predicted accuracy if we always predicted majority class (0)
- Imbalanced data (large majority class) $\rightarrow$ large NIR

Accuracy: Overall Classification Rate

- Good for general model performance
- Bad model: accuracy $<$ NIR

Sensitivity/Recall: True Positive Rate

- Strive for high sensitivity
- Will be **low** if there is class imbalance

Specificity: True Negative Rate

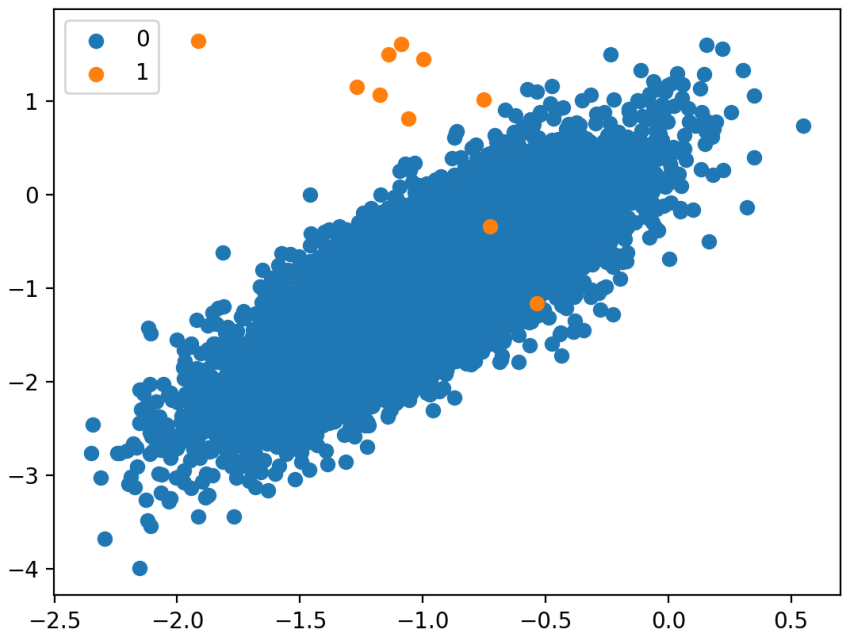
- Strive for high specificity
- Will be **high** if there is class imbalance

Class Imbalance

Main Idea: Unequal distribution of minority cases and majority cases in the training set.

- Occurs when minority class is rarer than majority class
- Example of Rare (Minority) Events: Interstate War, Financial Crises, Widget Defection

Figure 6: Extreme Class Imbalance Example



Degree of Class Imbalance

Degree of imbalance	Proportion of Minority Class
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	<1% of the data set

Class imbalance impacts model performance

- Predicting majority class every time $\rightarrow$ high No Information Rate (NIR)
- High NIR means model cannot improve on baseline performance that much
- Consequence on model performance:

Class imbalance impacts model performance

- Predicting majority class every time $\rightarrow$ high No Information Rate (NIR)
- High NIR means model cannot improve on baseline performance that much
- Consequence on model performance:
    - Models will have high specificity (TN) rates

Class imbalance impacts model performance

- Predicting majority class every time $\rightarrow$ high No Information Rate (NIR)
- High NIR means model cannot improve on baseline performance that much
- Consequence on model performance:
  - Models will have high specificity (TN) rates
  - Model will have low sensitivity (TP) rates (too few examples to learn)

Class imbalance impacts model performance

- Predicting majority class every time $\rightarrow$ high No Information Rate (NIR)
- High NIR means model cannot improve on baseline performance that much
- Consequence on model performance:
  - Models will have high specificity (TN) rates
  - Model will have low sensitivity (TP) rates (too few examples to learn)
  - If TP rate low enough, overall model accuracy may be less than NIR!

Significance: Imbalanced datasets $\rightarrow$ dangerously bad predictive models.

Correcting for class imbalance

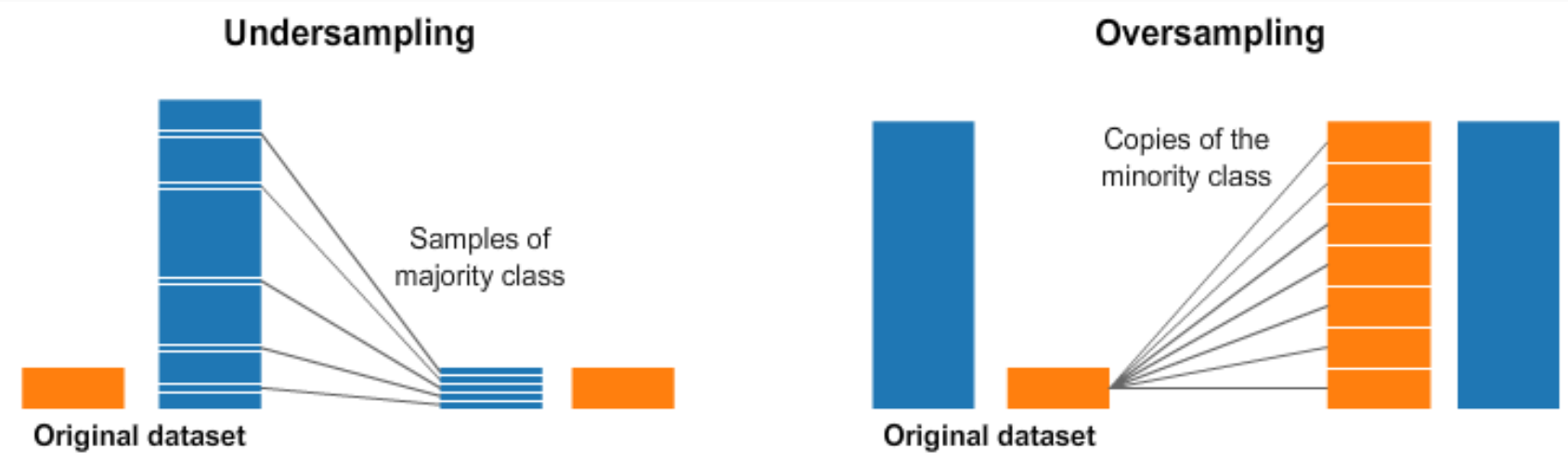
Solution to Class Imbalance: Rebalance the dataset so that majority and minority class are roughly equal

Techniques:

- Under-sampling (Down-sampling)
- Over-sampling (Up-Sampling)
- Synthetic Minority Oversampling Technique (SMOTE)

Under-Sampling and Over-Sampling

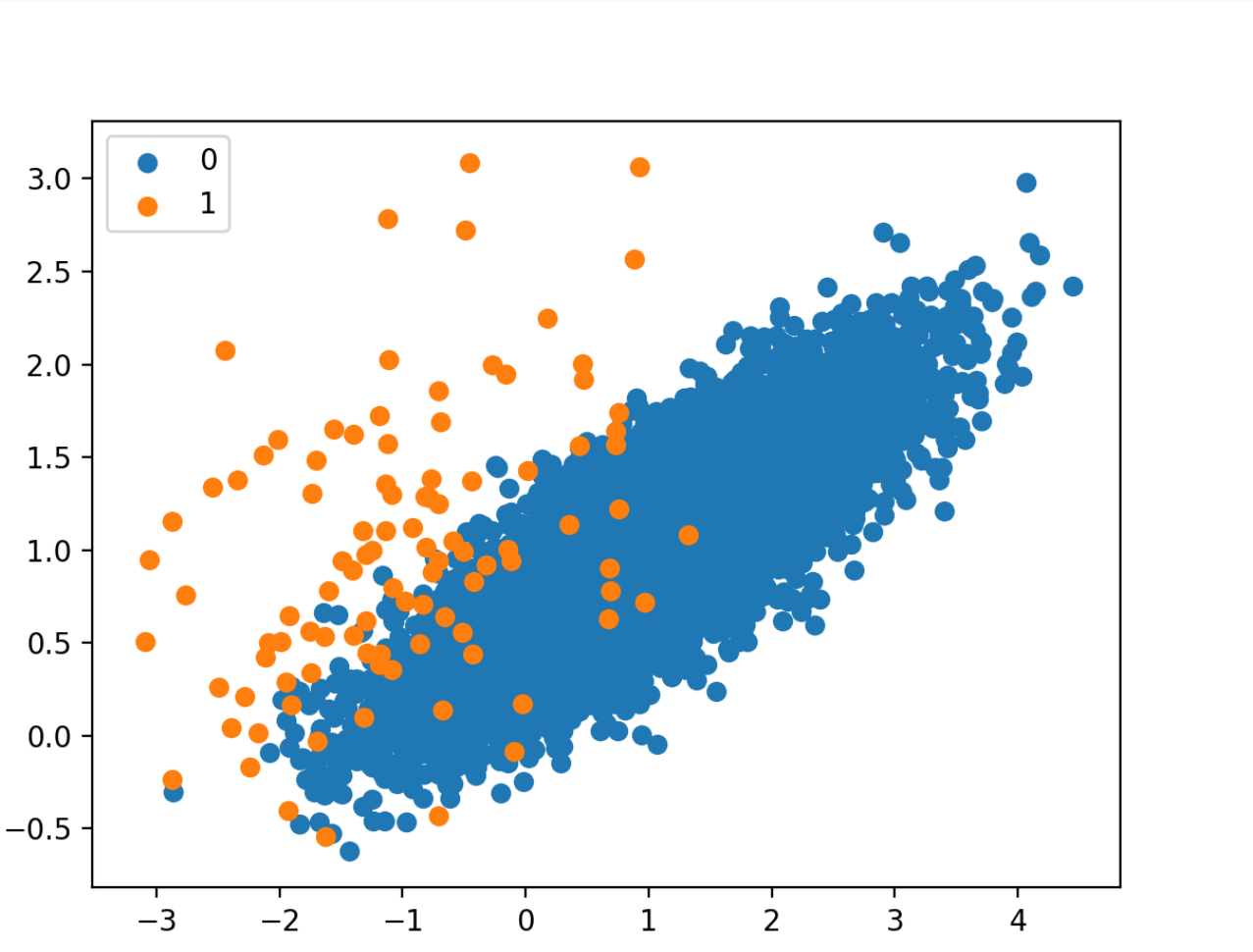
- **Under-sampling (down-sampling):** Keep minority class observations, randomly remove majority observations
- **Over-sampling (up-sampling):** Keep majority class observations, randomly duplicate minority observations



Synthetic Minority Oversampling Technique (SMOTE)

Creates synthetic observations based upon the existing minority observations using KNN

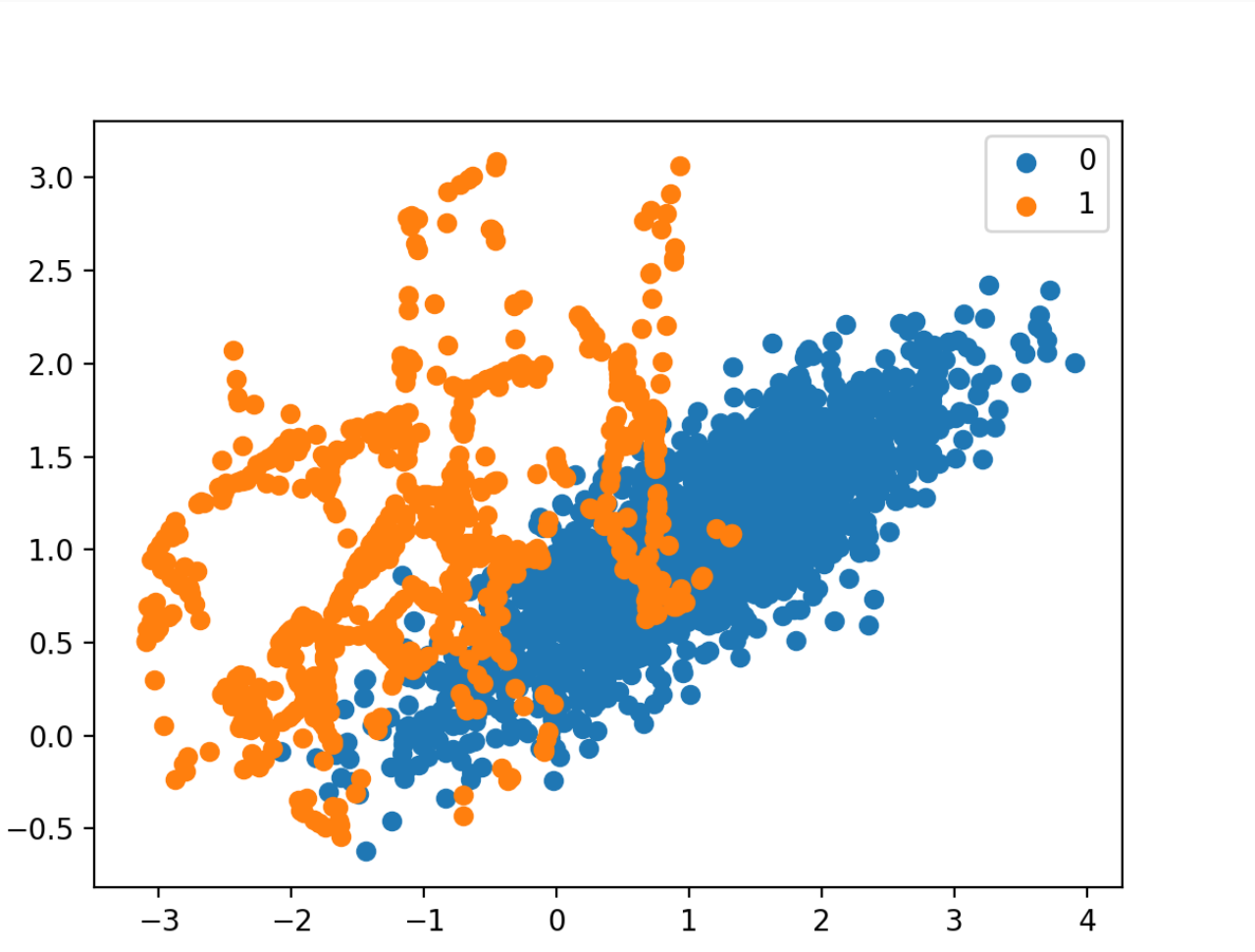
Figure 7: Original Data



Synthetic Minority Oversampling Technique (SMOTE)

Creates synthetic observations based upon the existing minority observations using KNN

Figure 8: Synthetic Observations



Selecting Best Class Imbalance Technique

General Rule: Oversampling generally performs the best, but to calibrate ...

Procedure:

1. Partition data into training and test set
2. Balance training data using different methods
3. Run model on each sampling method
4. Estimate recall rate (TP rate) for each model
5. Pick balanced dataset with highest recall

Conclusions

- Be aware of 7 deadly sins of political risk modeling
- Non-parametric models can be great if they pass the sniff test
- Random forests and GBM are great workhorse non-parametric models
- Class imbalance can impede model performance if not corrected