Advanced Tree-Based Methods

PSC 8185: Machine Learning for Social Science

Iris Malone

March 7, 2022

Materials adapted from Sergio Ballacado

Recap

Where We've Been:

- Non-parametric models 'black box' functional form
- Most common non-parametric models: KNN, CART, SVM
- CART top-down greedy algorithm produces high variance results

Recap

Where We've Been:

- Non-parametric models 'black box' functional form
- Most common non-parametric models: KNN, CART, SVM
- CART top-down greedy algorithm produces high variance results

New Terminology:

- Recursive Binary Splitting
- Pruning
- Gini Index/Gini impurity
- Variable Importance Plot
- Ensemble Method

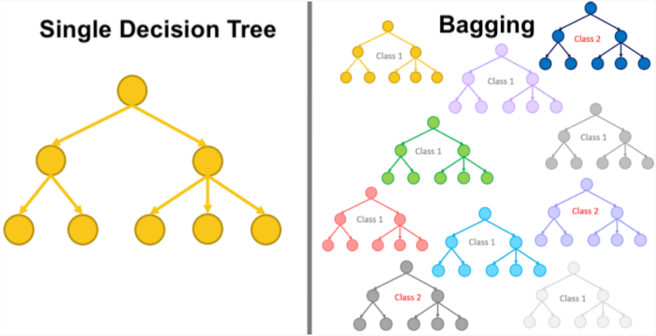
Agenda

1. Random Forests
2. Boosting
3. BART
4. Special Topic: Missing Data

Random Forests

Recap: Bagging popular alternative to decision trees

Bagging reduces high variance problem of CART 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 <u>same</u> number/type of predictors X_j

Problems with Bagging

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

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

Limits to Correlated Trees

Problem: Correlated trees produce potentially biased results …

- One highly influential predictor → prune all other predictors
- Collinear predictors → bias to variance-maximizing predictor
- Combination of continuous and binary measures → bias to continuous predictors

Limits to Correlated Trees

Problem: Correlated trees produce potentially biased results ...

- One highly influential predictor → prune all other predictors
- Collinear predictors → bias to variance-maximizing predictor
- Combination of continuous and binary measures → bias to continuous predictors

Solution: **Random Forests**

Random Forests

Main Idea: Create a large number of bootstrapped decision trees, but <u>vary the number of predictors</u> 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

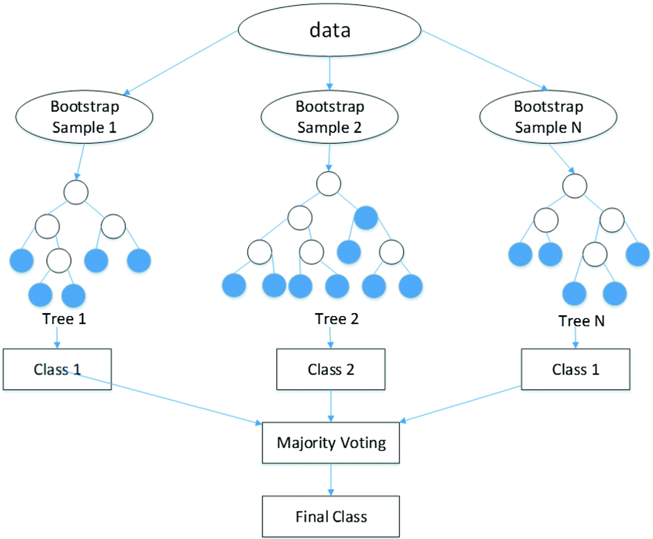
Random Forests

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 Forests

Make predictions by **majority class** voting:

Predictions with Random Forests

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 = Nationals, 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?

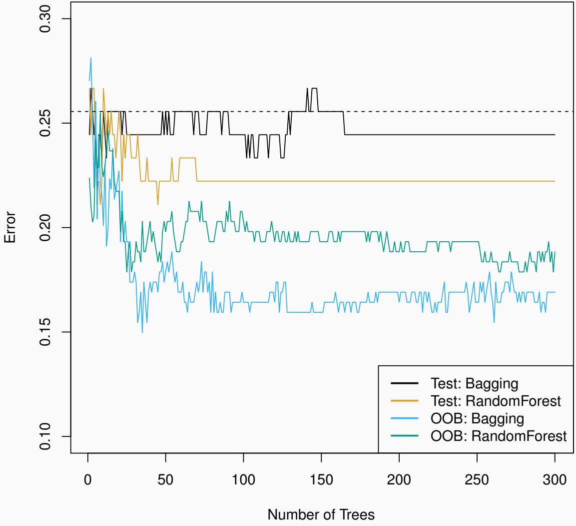
- Rule of Thumb: $\sim 500 - 1000$ trees
- Cross-Validation

How to choose the optimal m?

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

Comparison of Random Forest and Bagging by Number of Trees

Figure 1: RF tends to have lower validation error



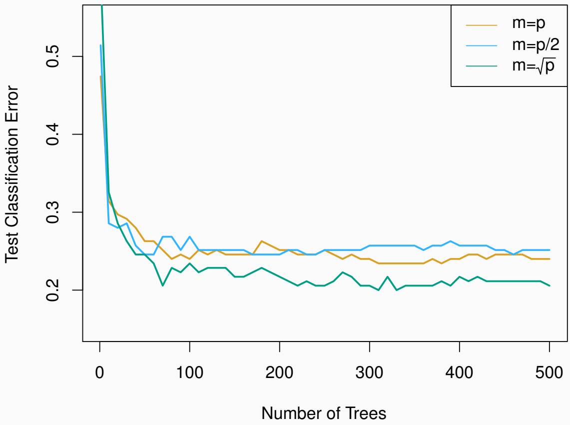
Random Forests vs Bagging

Note: Random forest is a special case of bagging (!)

- Bagging: $m = p$
- Random Forest: $m = \sqrt{p}$

Comparison of Random Forests and Bagging by $m \in [0, p]$

Figure 2: $m < p$ tends to results in lower validation error



Random Forest Evaluation for Classification Problems

Standard Metrics:

- Accuracy
- Kappa
- ROC

New Metrics:

- “Brier Score”
- Expected Percentage of Correct Predictions (ePCP)
- Separation Plot

Brier Score

Brier Score measures model performance for multi-categorical outcomes.

$$\text{Brier} = \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{R}(\hat{y}_{ti} - y_{ti})^2$$

- N is the overall number of classes; R is the number of possible classes
- Interpretation:
  - Brier score is range [0, 1]
  - Lower Brier scores = better performance
  - Does not tell you whether you predicted class accuracy (see PS 3)

Expected Percentage of Correct Predictions

Expected Percentage of Correct Predictions (epCP) is essentially the balanced accuracy of the model

$$ePCP = \frac{1}{N}(\sum_{y_i=1} \hat{y} + \sum_{y_i=0}(1 - \hat{y}))$$

Separation Plot

A **separation plot** is a popular visual tool of a model's predictive power

- Tells us extent to which model's predicted probability maps onto actual outcome
- Easy to visualize FP vs TP (like ROC)
- Easy to visualize sparsity of data and class distribution

See Greenhill et al. (2011) "The Separation Plot: A New Visual Method for Evaluating the Fit of Binary Models" for more.

Motivating Example: Predict War and Peace

Have observations $\{A, B, C, D, E, F\}$

	Predicted War	Predicted Peace
Actual War	{C, E}	{F}
Actual Peace	{A, D}	{B}

Motivating Example: Predict War and Peace

Table 1 Sample Data

Country	Actual Outcome (y)	Fitted Value ($\hat{p}$)
A	0	0.774
B	0	0.364
C	1	0.997
D	0	0.728
E	1	0.961
F	1	0.422

Brier Score

Table 3 Calculation of Brier Scores

Country	Actual Outcome (y)	Fitted Value ($\hat{p}$)	Brier Score $(\hat{p} - y)^2$
A	0	0.774	0.599
B	0	0.364	0.132
C	1	0.997	0.000
D	0	0.728	0.530
E	1	0.961	0.002
F	1	0.422	0.334

Separation Plot

TABLE 4 Rearrangement (and Coloring) of the Data Presented in Table 1 for Use in the Separation Plot

Country	Fitted Value ($\hat{p}$)	Actual Outcome (y)
B	0.364	0
F	0.422	1
D	0.728	0
A	0.774	0
E	0.961	1
C	0.997	1

Sort by $\hat{p}$ and color code them: red if event happened and tan if no event happened

Separation Plot

FIGURE 2 Separation Plot Representing the Data Presented in Table 1

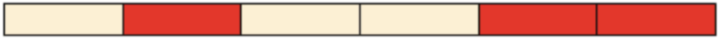
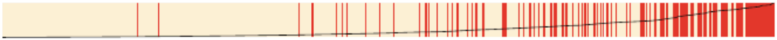


FIGURE 3 Separation Plot for a Larger Data Set

Take 5 observations sorted by $\hat{p}$ and color code by whether event actually happened

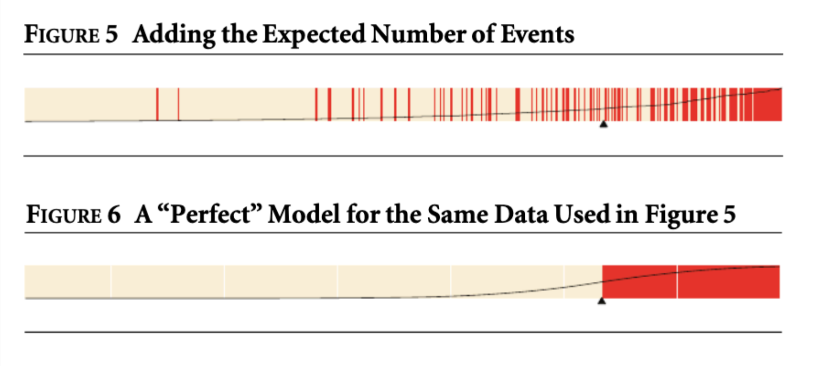
Separation Plot

Figure 4 Adding a Graph of $\hat{p}$ to the Separation Plot



We can expand for larger data set and add black line for $\hat{p}$. Can now compare events versus predicted probabilities.

Separation Plot



If model was perfect, we'd see complete event color-coded separation

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 → slow and computationally expensive
- Does not perform as well as other algorithms
- Top-down approach → suboptimal splits
- Assumes independence between observations → 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

Use **gradient descent algorithm** to minimize error

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:

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

    - Update the residuals:

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

- 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

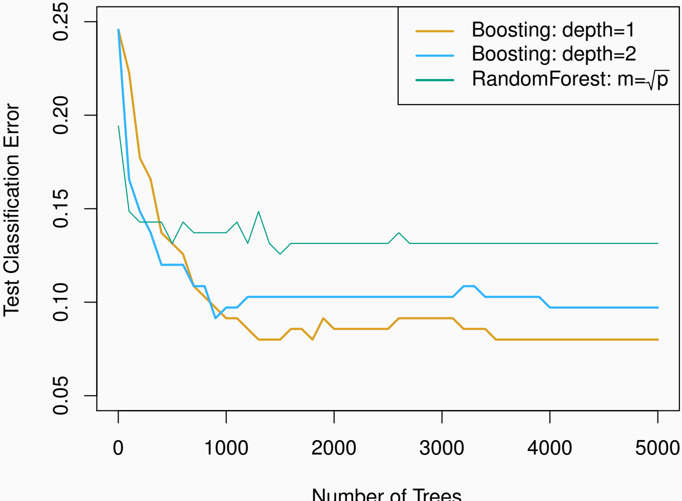
- involves bootstrap sampling → independence between trees
- no learning
- high risk of overfitting

GBM

- does not involve bootstrap sampling → dependence between trees
- slow learning
- less risk of overfitting

GBM vs Random Forests

Boosting tends to perform better than random forest



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

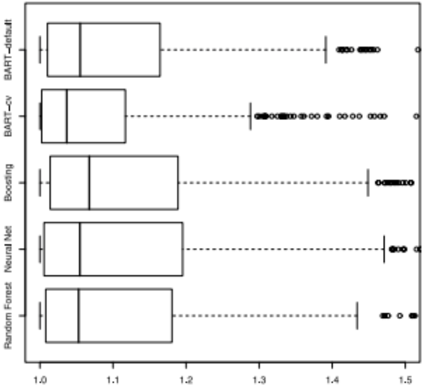


FIG. 2. *Boxplots of the RRMSE values for each method across the* 840 *test/train splits. Percentage RRMSE values larger than* 1.5 *for each method (and not plotted) were the following: random forests* 16.2%, *neural net* 9.0%, *boosting* 13.6%, *BART-cv* 9.0% *and BART-default* 11.8%. *The Lasso (not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

Figure 3: 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 → 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

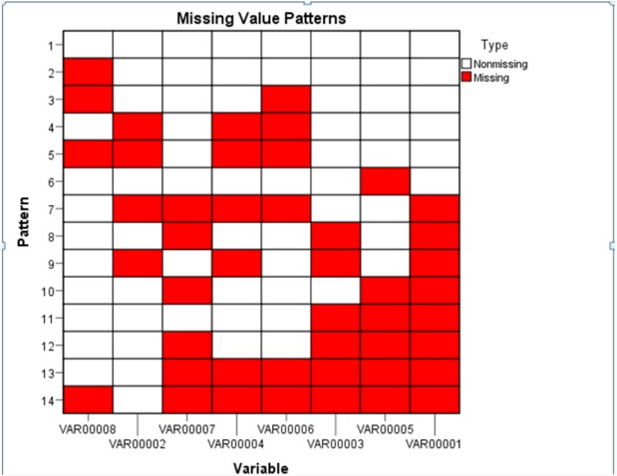
- **Local Distribution:** Identifies a variable x as relevant if its inclusion proportion falls above the $1 - \alpha$ quantile of the permutation distribution for x.
- **Global Distribution:** Stricter approach; identifies a variable x as relevant if its inclusion proportion falls above the $1 - \alpha$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

See Bleich et al (2014), p. 761 for example.

Special Topic: Missing Data

Problem of Missing Data

Motivation: Observational data often has missing data. When there is a large number of predictors, higher likelihood you might be missing at least one predictor.



Types of Missing Data

1. **Missing Completely at Random (MCAR):** No systematic pattern in which observations are missing

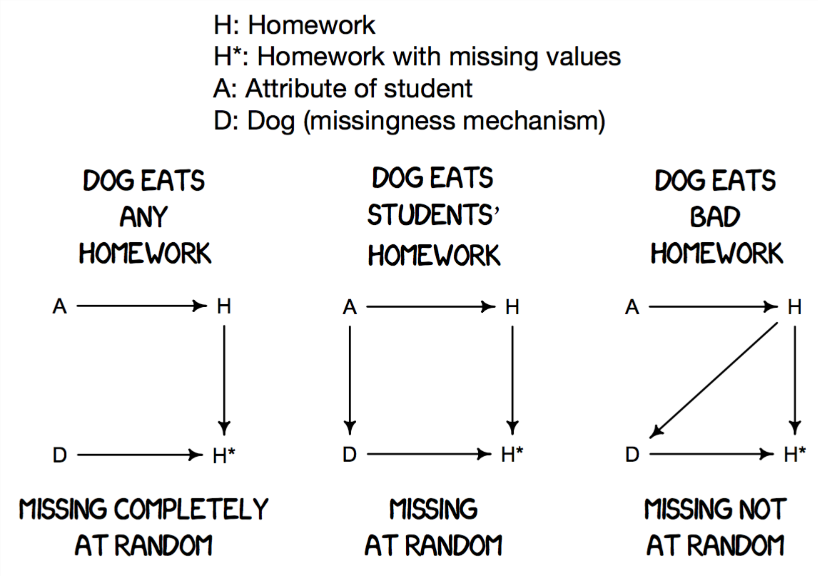
Types of Missing Data

1. **Missing Completely at Random (MCAR):** No systematic pattern in which observations are missing
2. **Missing at Random (MAR):** Observations are missing for some values, but not due to a specific attribute (missingness is conditional on a separate attribute)

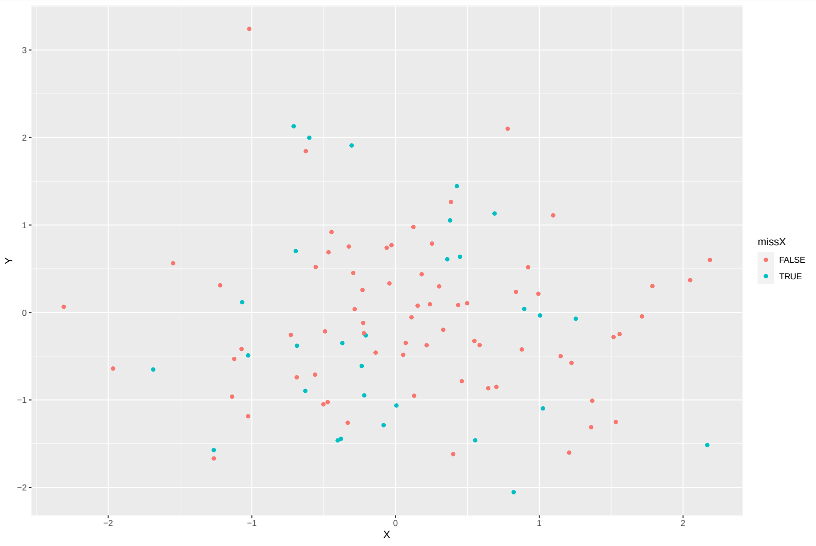
Types of Missing Data

1. **Missing Completely at Random (MCAR):** No systematic pattern in which observations are missing
2. **Missing at Random (MAR):** Observations are missing for some values, but not due to a specific attribute (missingness is conditional on a separate attribute)
3. **Missing Not at Random (MNAR):** Observations are missing for some values as a function of a particular attribute/mechanism

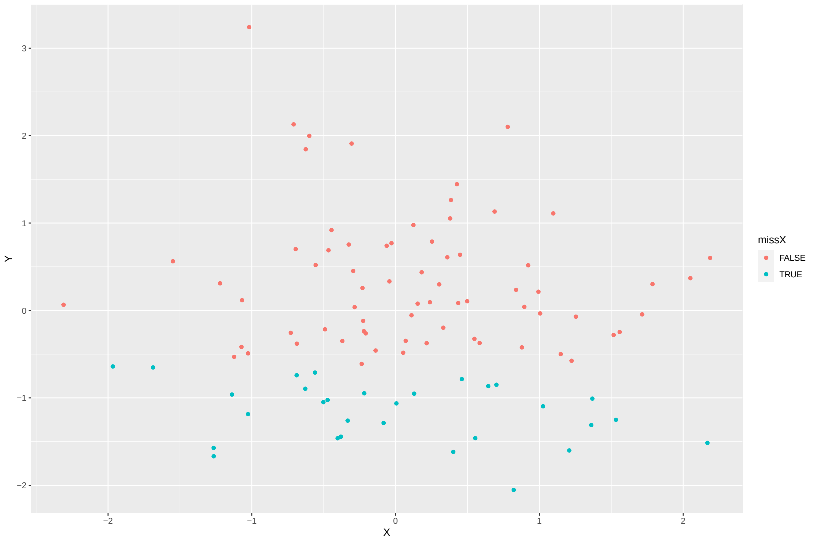
Example of Missing Data



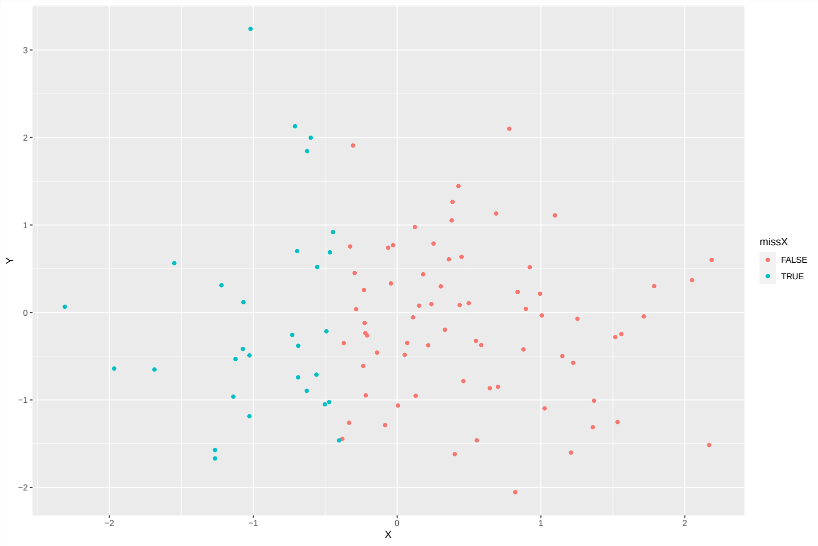
Missing Completely at Random



Missing (Conditionally) at Random



Missing Not at Random



Consequences of Missing Data

Three Problems with Missing Data:

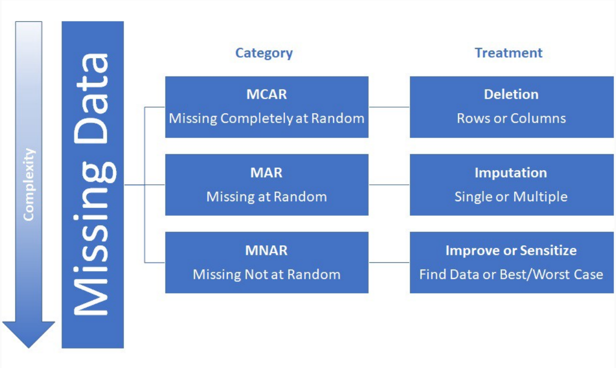
- Less learning power (fewer n)
- Selection bias (less representative)
- Omitted variable bias (biased estimates)

Main Takeaway: <u>Never</u> omit missing missing observations without understanding what type of missing data you have.

Solutions to Missing Data

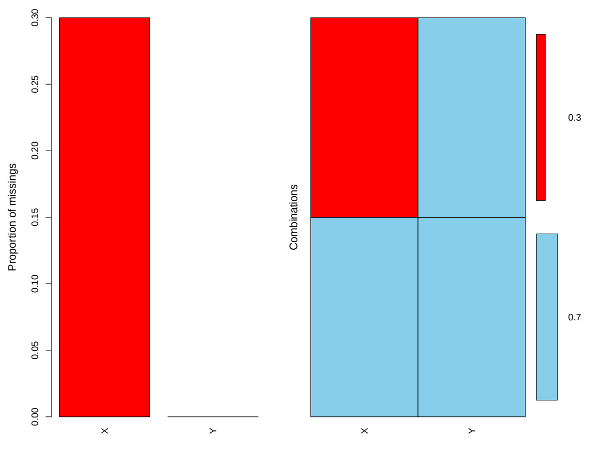
1. Delete Missing Observations
2. Ignore Missing Observations
3. Impute Missing Observations
4. Improve Missing Observations

Missing Data Type Governs Solution



Visualizing Missing Data

aggr(df, prop = T, numbers = T)



Solutions to Missing Data

1. Delete Missing Observations
2. Ignore Missing Observations
3. Impute Missing Observations
4. Improve Missing Observations

Deletion Methods

If data is MCAR ...

- **Listwise Deletion:** Delete all rows where one or more values are missing.

Deletion Methods

If data is MCAR ...

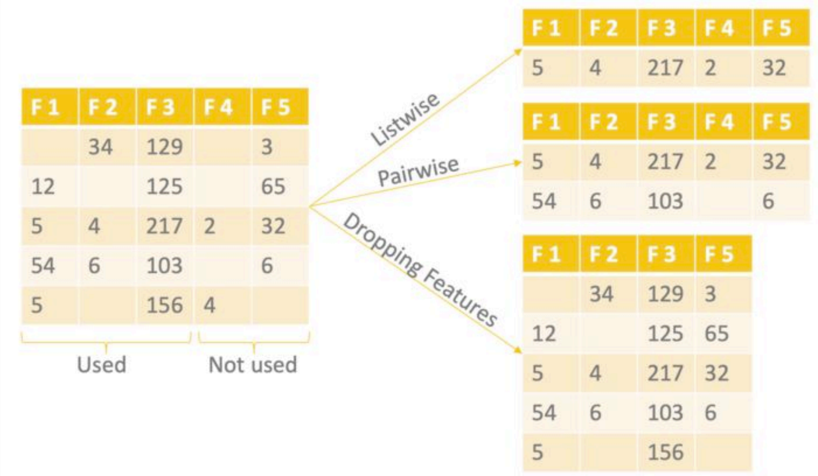
- **Listwise Deletion:** Delete all rows where one or more values are missing.
- **Pairwise Deletion:** Delete only the rows that have missing values in the columns used for the analysis.

Deletion Methods

If data is MCAR ...

- **Listwise Deletion:** Delete all rows where one or more values are missing.
- **Pairwise Deletion:** Delete only the rows that have missing values in the columns used for the analysis.
- **Dropping Features:** Drop entire columns with more missing values than a given threshold, e.g. 60

Deletion Methods



F1	F2	F3	F4	F5
	34	129		3
12		125		65
5	4	217	2	32
54	6	103		6
5		156	4	

F1	F2	F3	F4	F5
5	4	217	2	32

F1	F2	F3	F4	F5
5	4	217	2	32
54	6	103		6

F1	F2	F3	F5
	34	129	3
12		125	65
5	4	217	32
54	6	103	6
5		156	

Solutions to Missing Data

1. Delete Missing Observations
2. **Ignore Missing Observations**
3. Impute Missing Observations
4. Improve Missing Observations

Ignore Methods

If data is MCAR for only some observations, then you may alternatively ignore by passing through the data (na.action=na.pass, na.action=na.fail)

Solutions to Missing Data

1. Delete Missing Observations
2. Ignore Missing Observations
3. **Impute Missing Observations**
4. Improve Missing Observations

Imputation Methods

If data is MAR (and it often is), then you may impute using:

- Zero/Constant Values
- Mean or Median Values
- KNN Values
- Multivariate Imputed Chained Equations (MCMC)
- Deep Learning Imputation

Zero/Constant Imputation

Method: replaces the missing values with either zero or any constant value you specify (often mode)

	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN
1	9	NaN	9.0	0	7.0
2	19	17.0	NaN	9	NaN

df.fillna(0) →

	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	0.0
1	9	0.0	9.0	0	7.0
2	19	17.0	0.0	9	0.0

Problem: Can skew results depending on input value.

Mean or Median Imputation

Method: Calculate the mean/median of the non-missing values in a column and then replacing the missing values within each column separately and independently from the others

	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN
1	9	NaN	9.0	0	7.0
2	19	17.0	NaN	9	NaN

mean()

	col1	col2	col3	col4	col5
0	2.0	5.0	3.0	6.0	7.0
1	9.0	11.0	9.0	0.0	7.0
2	19.0	17.0	6.0	9.0	7.0

Advantages and Disadvantages to Mean Imputation

Advantages:

Disadvantages:

Advantages and Disadvantages to Mean Imputation

Advantages:

- Easy and fast.
- Works well with small numerical datasets.

Disadvantages:

Advantages and Disadvantages to Mean Imputation

Advantages:

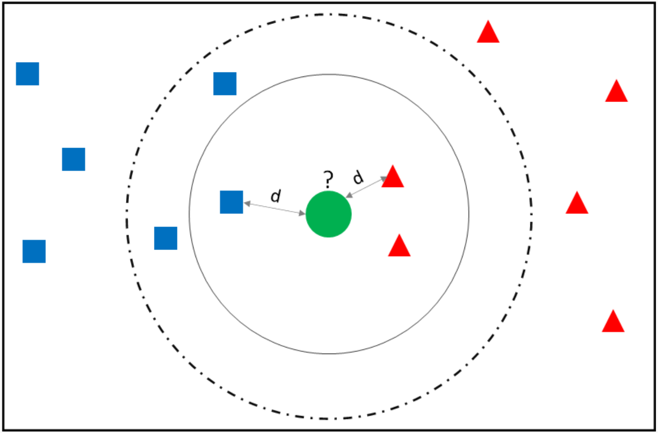
- Easy and fast.
- Works well with small numerical datasets.

Disadvantages:

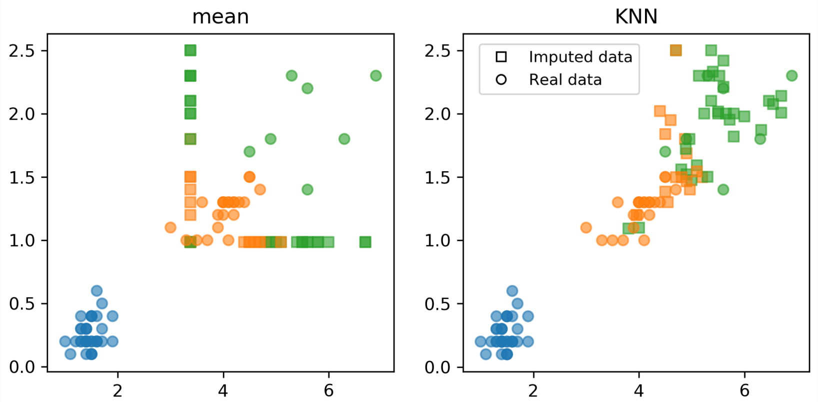
- Doesn’t factor the correlations between features. It only works on the column level.
- Will give poor results on encoded categorical features
- Not very accurate.
- Doesn’t account for uncertainty in the imputations

KNN Imputation

Method: Finding the k's closest neighbours to the observation with missing data and then imputing those values based on the non-missing values in the neighborhood.



Imputation Example



Advantages and Disadvantages to KNN Imputation

Advantages:

Disadvantages:

Advantages and Disadvantages to KNN Imputation

Advantages:

- Can be much more accurate than the mean, median or most frequent imputation methods
- Requires very few assumptions

Disadvantages:

Advantages and Disadvantages to KNN Imputation

Advantages:

- Can be much more accurate than the mean, median or most frequent imputation methods
- Requires very few assumptions

Disadvantages:

- Computationally expensive. Requires storing the whole training dataset in memory.
- Sensitive to outliers in the data

Solutions to Missing Data

1. Delete Missing Observations
2. Ignore Missing Observations
3. Impute Missing Observations
4. **Improve Missing Observations**

Solutions to Missing Data

1. Delete Missing Observations
2. Ignore Missing Observations
3. Impute Missing Observations
4. **Improve Missing Observations** → Collect more data!

Conclusion

- Random Forest improves over bagging by only examining some predictors at a time
- Boosting and BART improves over CART, bagging, and RF by sequentially growing trees
- Slow learning methods → better model performance
- Imputation can resolve data when it is MAR