

# Ensemble Methods and Blending

STAT440/24

**Dr. Lloyd T. Elliott**

2023/12/04

# Ensemble methods

Combine predictions of multiple individual models (through aggregation we increase strength, reduce weakness)

- Bagging. Train multiple instances of the same base model on random subsets of training data, averaging output. Example: Random Forest
- Boosting. Iterative method. Models are trained sequentially, with each model focusing on examples that previous models struggled with. Example: XGBoost
- Stacking. Models are trained in parallel, and then fed into a "meta-model"

Banachewicz & Massaron Chapter 9

# Bagging: Random forests

- Ensemble method  $f(x) = \frac{1}{m} \sum_{j=1}^m g(x; \theta_j)$
- Mean of  $m$  decision trees, each trained on a subset of the data (subset data items, subset features; Breiman 2001)

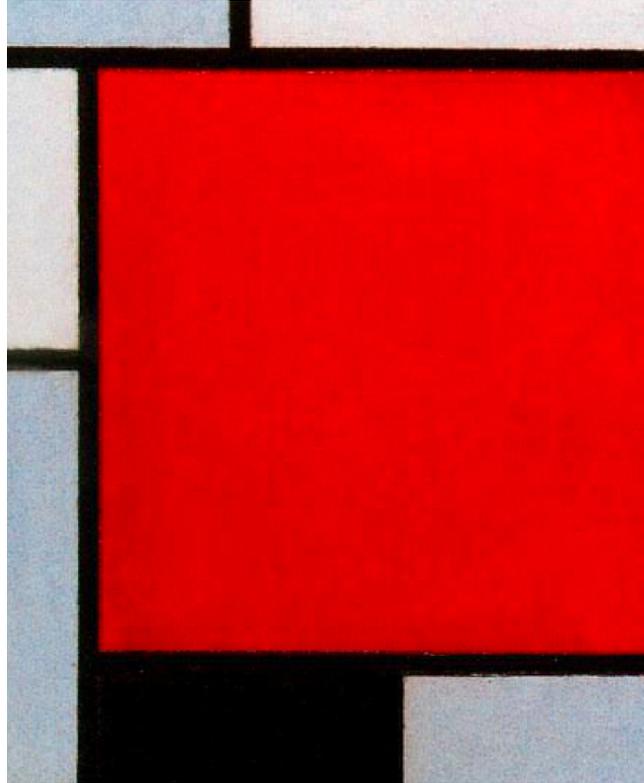
# Random forests

- Strengths:
  - Training is fast, good performance on "tall" data (many data items); through parallel processing & subsetting
  - Not so susceptible to overfitting
  - Good performance on complex (non-linear) data, without feature engineering
- Weaknesses:
  - Hard to visualize/summarize
  - Poor performance on high dimensional or sparse data (e.g., text)
- Best R package: *randomForest*

# Random forest algorithm

- Train  $m$  decision trees on subsets of the data:
  - For each  $m$ , choose a random subset of data items and a random subset of features, and restrict  $\mathbf{X}$  to this subset
  - Train decision tree  $f(\cdot, \theta_j)$  on the subset where  $\theta_j$  is the  $j$ -th decision tree.
- Predict using the mean:
  - Prediction for  $y$  is  $f(x) = \frac{1}{m} \sum_{j=1}^m g(x; \theta_j)$  or  $f(x) = \operatorname{argmax}_y \#\{j : g(x; \theta_j) = y\}$

# Decision trees



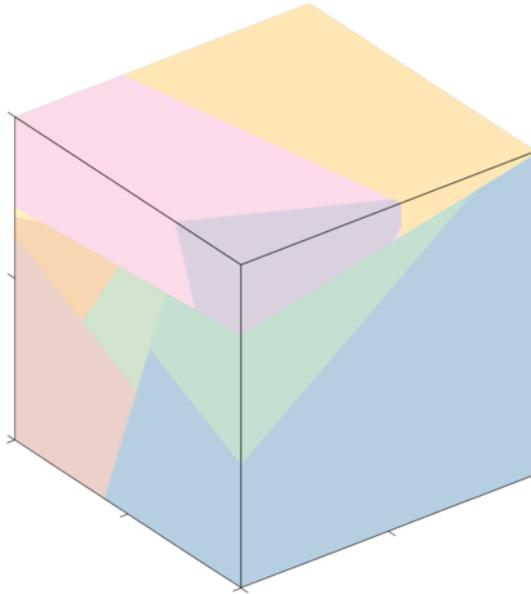
- Recursively split data along a dimension at a value that minimizes a loss function

# Decision trees (pseudocode)

- Consider the root node of the tree, using all available data
  - To consider a node, loop through every feature of the data, and every possible split point of the data used at that node
  - Each potential split implies a predictions for the two subsets, and a loss. Choose the split that minimizes the loss
  - If a stopping criteria is not met, then this node involves splitting on the feature at the split point that minimizes the loss. Then, recurse by considering two new daughter nodes for this node, using the two subsets of the data implied by the split for this node
  - If a stopping criteria is met, then this node is a leaf node. Output values for test data items reaching this node are implied by the predictions for this subset

# Decision trees (non-axis aligned)

- Extension of decision trees to non-axis aligned cuts (Ge 2019)



# Boosting: XGBoost

A boosting algorithm for decision trees, with many enhancements:

- Shallow trees (improves resilience to overfitting)
- L1/L2 regularization
- Cross validation and early stopping

Chen & Guestrin. XGBoost: A scalable tree boosting system. KDD 2016

# Boosting: XGBoost

```
library(xgboost)
data = iris
X = as.matrix(data[, 1:4])
Y = as.numeric(data$Species) - 1
model = xgboost(data = X, label = Y, objective = "multi:softprob",
  num_class = 3, nrounds = 100)
train = predict(model, newdata = X, reshape = TRUE)
```

```
cat(sprintf("Training accuracy: %0.1f\n", mean(max.col(train,
  "first") - 1 == Y) * 100))
```

Training accuracy: 100.0

# XGBoost theory

The XGBoost algorithm works as follows:

1. Consider a series of models  $M_1, M_2, \dots$
2. Train the first model  $M_1$  on the training set
3. Compute the residuals  $r_1$  for the first model on the training set
4. Train the  $i$ -th model to predict the residuals  $r_{i-1}$
5. Compute the residuals  $r_i$  for the  $i$ -th model on the residuals  $r_{i-1}$
6. Repeat 4/5 until a stopping condition is met
7. Make the final prediction: The sum of the predictions of all the models

# Stacking

1. Consider a series of models  $M_1, \dots, M_k$ , and suppose we have a training/validation/testing split
2. Train each model  $M_1$  on the training set
3. Compute the predictions of each model  $\hat{y}_i$  on the validation set
4. Perform a multilinear regression for the validation targets  $y$  against the predictions of each model  $(\hat{y}_1 \dots \hat{y}_k)$ , yielding regression weights  $w_1, \dots, w_k$
5. Compute the predictions of each model  $\tilde{y}_i$  on the testing set
6. Make the final prediction:  $\sum_{i=1}^k w_i \tilde{y}_i$
7. Repeat 4/5 until a stopping condition is met
8. Make the final prediction: The sum of the predictions of all the models