GR5260
Programming for Quantitative & Computational Finance

Instructor: Ka Yi Ng

April 21, 2023

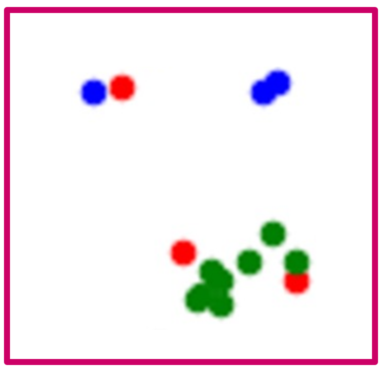
Tree-based models

Tree-based ML models

- Partitioning feature space into rectangular blocks, assigning a constant prediction value to each block
- Decision trees using CART algorithm
- Random forests: ensemble example
- Recall:
  - Training dataset: $(\boldsymbol{x}^{(\mathbf{1})}, \boldsymbol{y}^{(\mathbf{1})}), \ldots, (\boldsymbol{x}^{(\boldsymbol{m})}, \boldsymbol{y}^{(\boldsymbol{m})})$
  - Features: $\boldsymbol{x} = (x_1, \ldots, x_n)$
  - Classification: $\boldsymbol{y} \in \{1,2, \ldots, K\}$
  - Regression: $\boldsymbol{y} \in \mathbb{R}$

Classification: CART algorithm

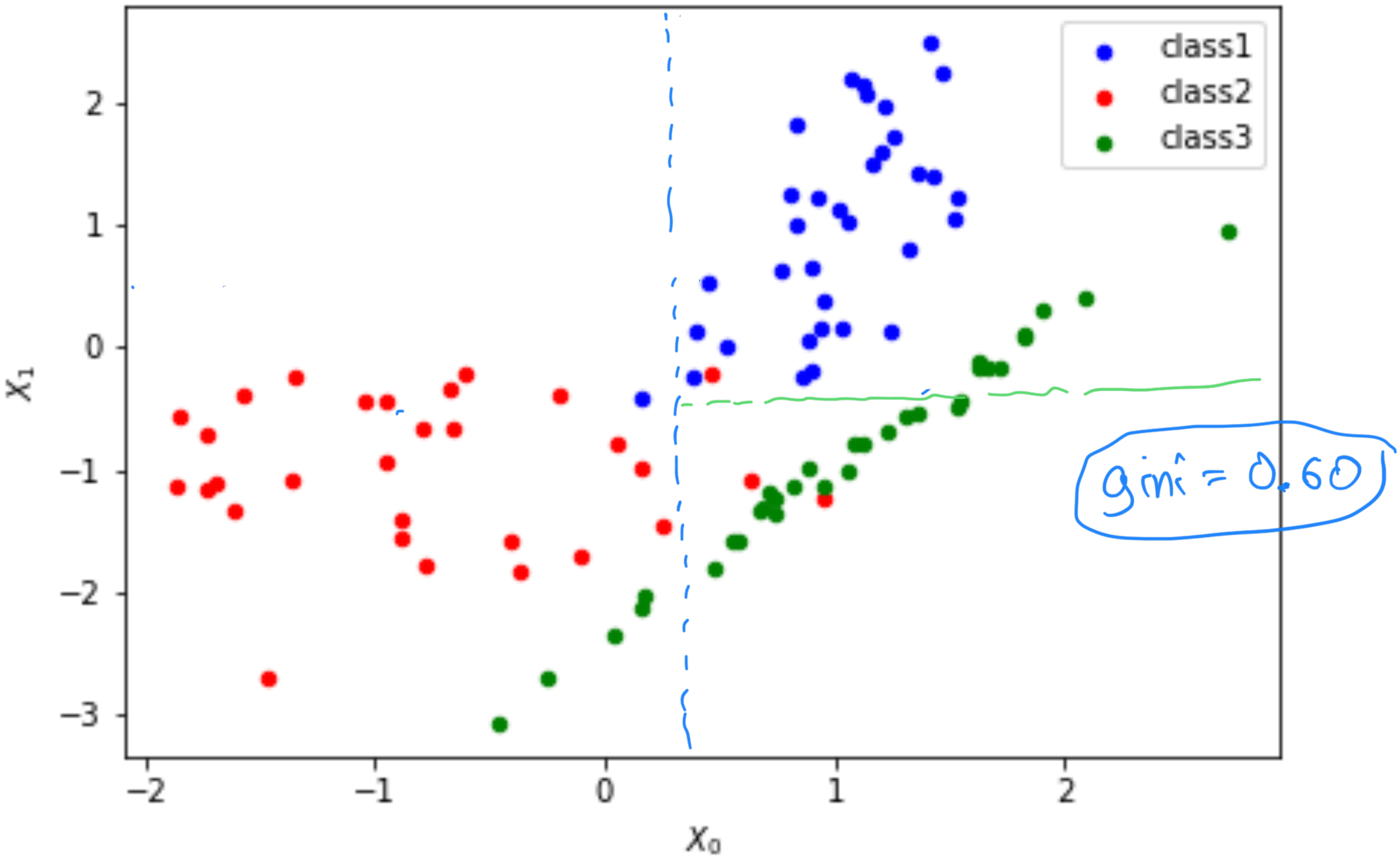
- Gini impurity index for a set of N points in K classes:
- $$\boldsymbol{G} = \underbrace{\sum_{\boldsymbol{i}=1}^{\boldsymbol{K}} \widehat{\boldsymbol{p}}_{\boldsymbol{i}}(\mathbf{1} - \widehat{\boldsymbol{p}}_{\boldsymbol{i}})}, \qquad \widehat{\boldsymbol{p}}_{\boldsymbol{i}} = \frac{\boldsymbol{N}_{\boldsymbol{class\_i}}}{\boldsymbol{N}}$$ and $\boldsymbol{N}_{\boldsymbol{class\_i}}$ = # points of class $\boldsymbol{i}$
- As an error measure $\hat{p}_{blue}(1-\hat{p}_{blue})$



- 3 red points, 3 blue points, 7 green points
- $G = \left(\frac{3}{13}\right)\left(\frac{10}{13}\right) + \left(\frac{3}{13}\right)\left(\frac{10}{13}\right) + \left(\frac{7}{13}\right)\left(\frac{6}{13}\right) = 0.60$
- 3 red points, 4 blue points, 6 green points
- $G = \left(\frac{3}{13}\right)\left(\frac{10}{13}\right) + \left(\frac{4}{13}\right)\left(\frac{9}{13}\right) + \left(\frac{6}{13}\right)\left(\frac{7}{13}\right) = 0.64$

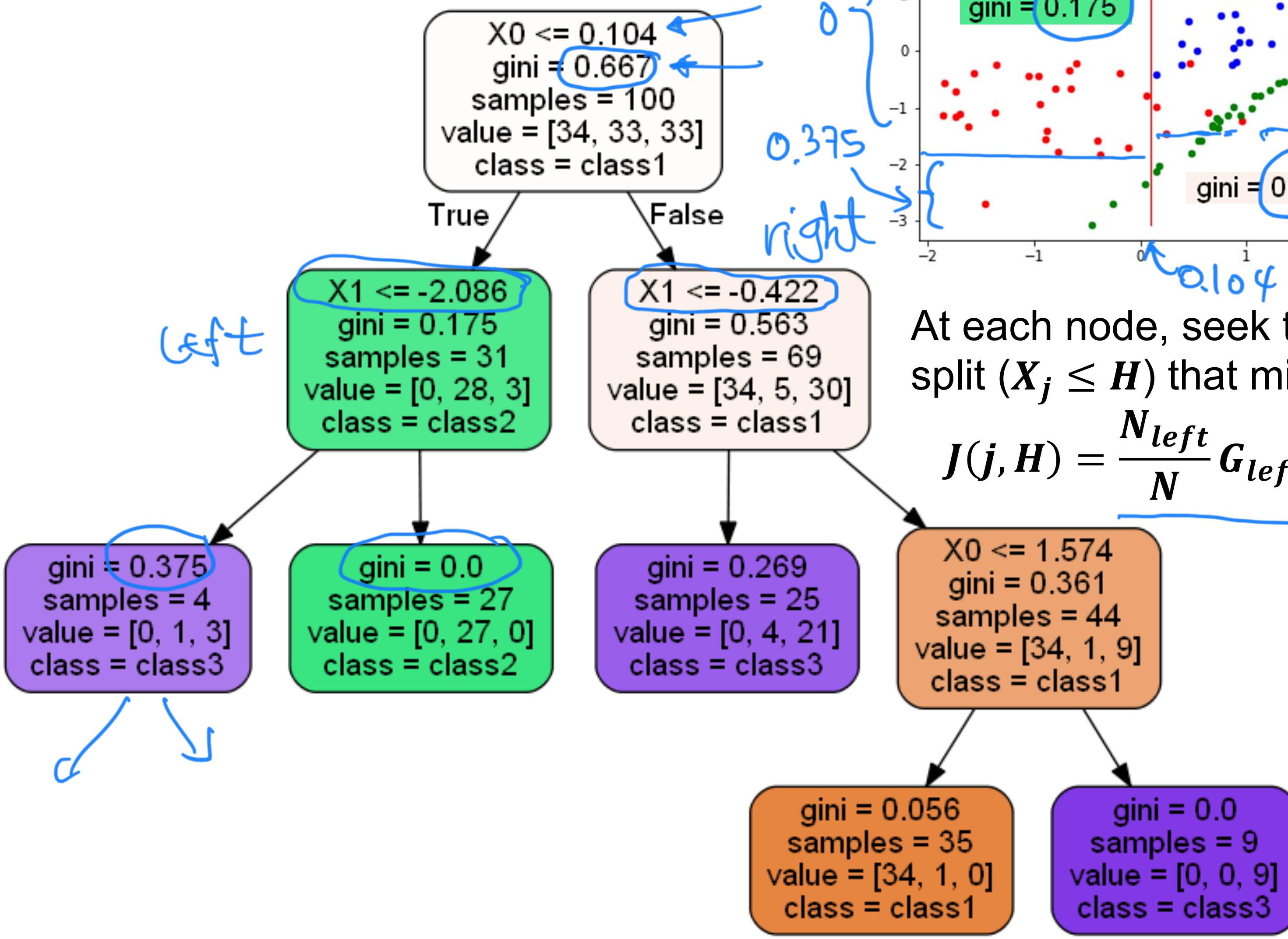
Decision Tree: Classification

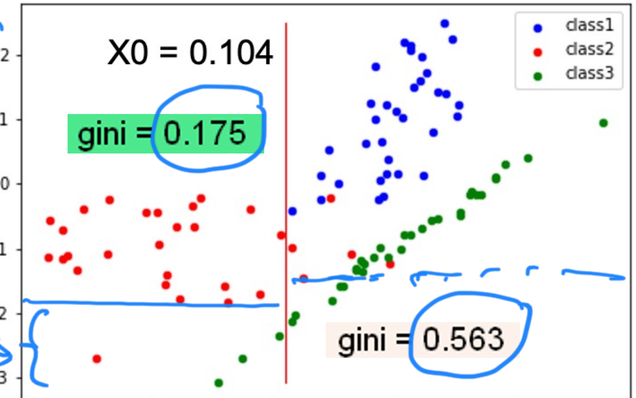
Partition n-dim feature space using hyperplanes defined by feature values



Classification: CART algorithm

- <u>Gini impurity index</u> $G = \sum_{i=1}^{K} \hat{p}_i(1 - \hat{p}_i)$





At each node, seek the best feature split ($X_j \leq H$) that minimizes:

$$J(j, H) = \frac{N_{left}}{N} G_{left} + \frac{N_{right}}{N} G_{right}$$

Classification: CART algorithm

- At each node:
  - Compute its Gini impurity index, $\boldsymbol{G_{node}}$
  - Among all features, find the feature $(\boldsymbol{X_j})$ and threshold level $\boldsymbol{H}$ that gives the best split $(\boldsymbol{X_j \leq H})$ ie. minimizes the cost function

$$\boldsymbol{J(j, H) = \frac{N_{left}}{N} G_{left} + \frac{N_{right}}{N} G_{right}}$$

  - If the difference $\boldsymbol{G_{node} - J(\hat{j}, \widehat{H}) > \delta}$, split the node using the best split criterion; otherwise, no split
- Repeat the above until no nodes can be split or some predefined criterion (eg. #nodes, tree depth) is met
- Prediction:
  - For each leaf node: predicted class probability distribution
  - Given a new data point $\boldsymbol{x_{new}}$, determine the leaf node it belongs to.
  - Predicted label = class with highest probability in the leaf node

Regression: CART algorithm

- At each node:
  - $\bar{y}_{node}$ = mean of label values of all points belonging to the node

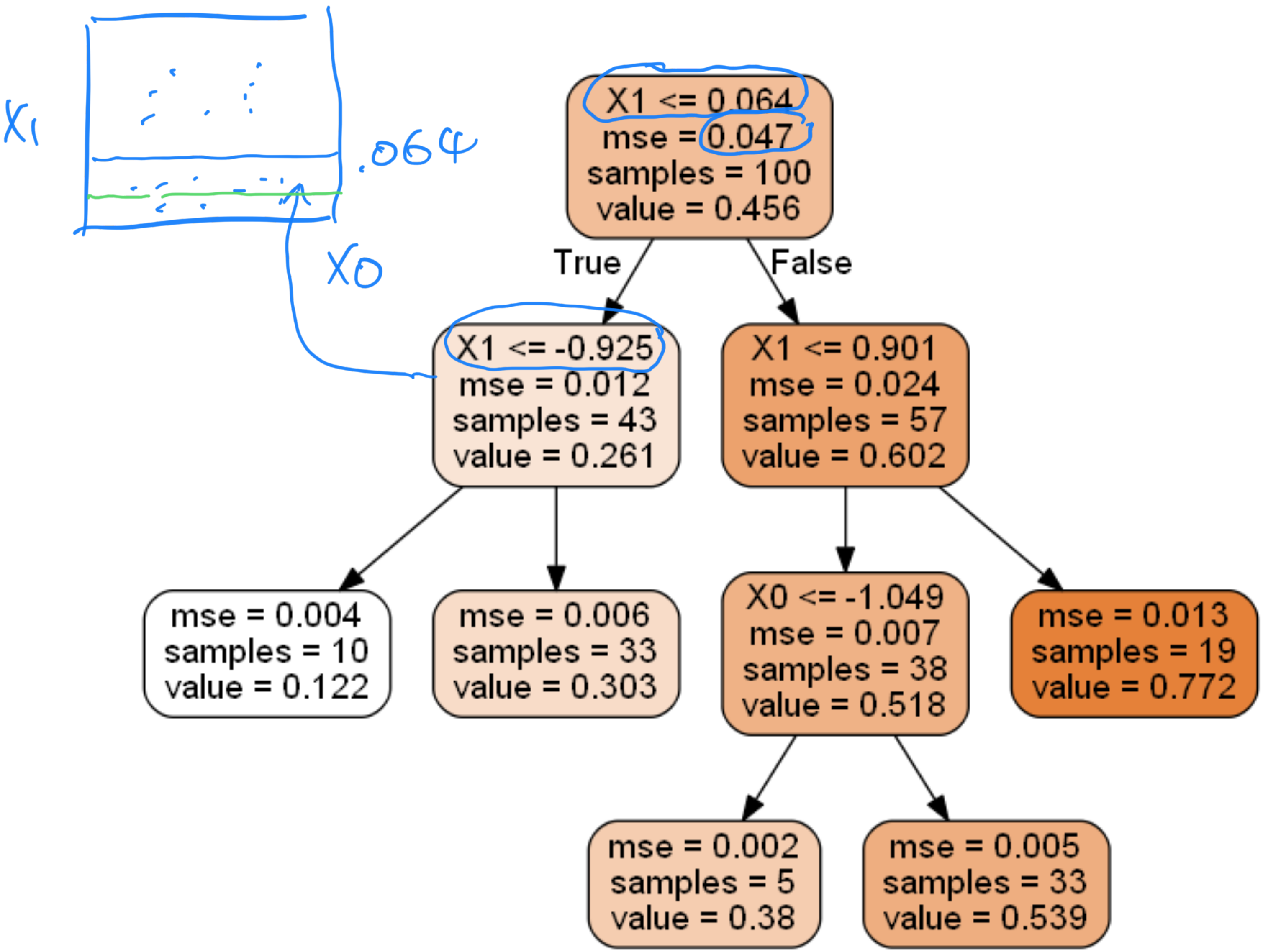
$$MSE_{node} = \frac{1}{N} \sum_{i \epsilon node} (y^{(i)} - \bar{y}_{node})^2$$

  - Among all features, find the feature (X_j) and threshold level H that gives the best split ($X_j \leq H$) ie. minimizes the cost function

$$J(j, H) = \frac{N_{left}}{N} MSE_{left} + \frac{N_{right}}{N} MSE_{right}$$

  - If the difference $MSE_{node} - J(\hat{j}, \widehat{H}) > \delta$, split the node using the best split criterion; otherwise, no split
- Repeat the above until no nodes can be split or some predefined criterion (eg. #nodes, tree depth) is met
- Prediction:
  - Given a new data point x_{new}, find the leaf node $\mathscr{N}$ it belongs to.
  - Predicted y_{new} = average label values of training points in node $\mathscr{N}$

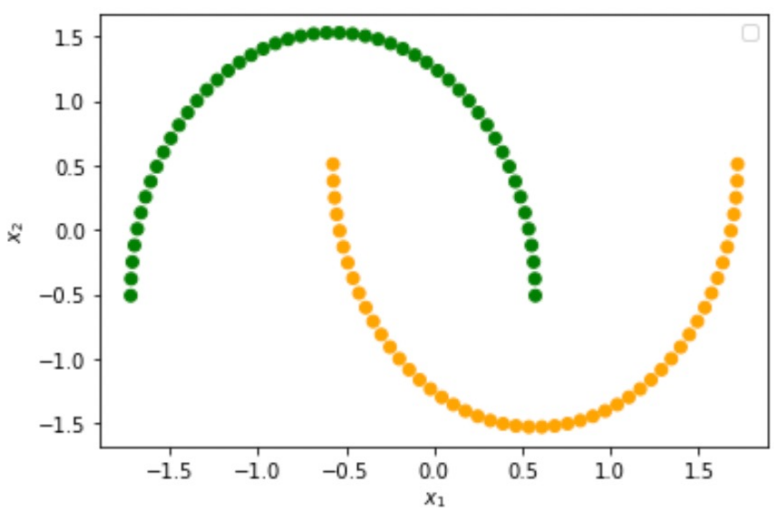
Regression: CART algorithm

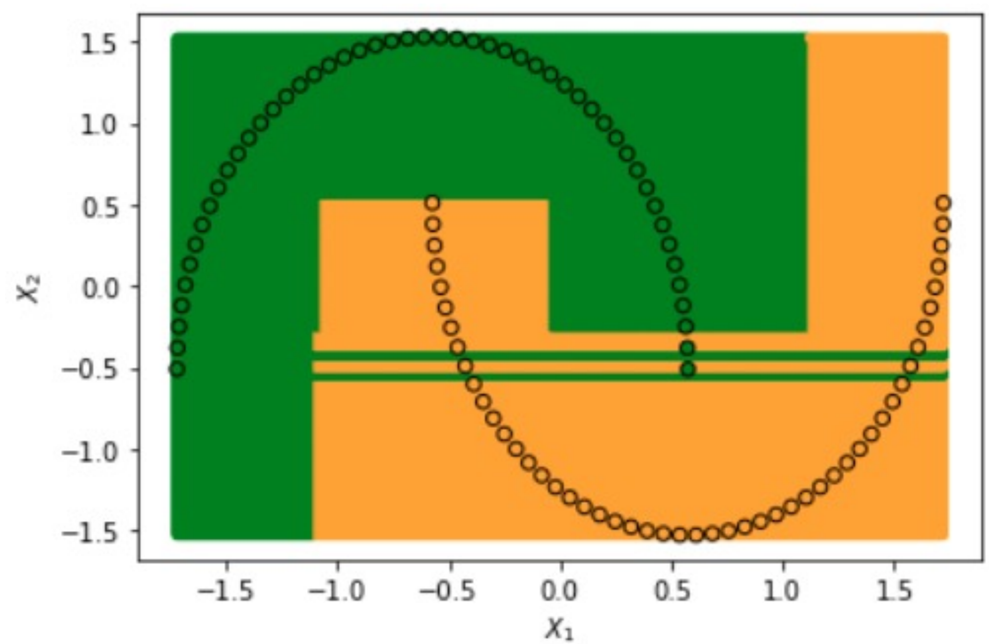


Benefits & Limitations

- Benefits:
  - Interpretability: explicit descriptive rules
  - Flexible, no model function assumptions
- Limitations:
  - Prone to overfitting

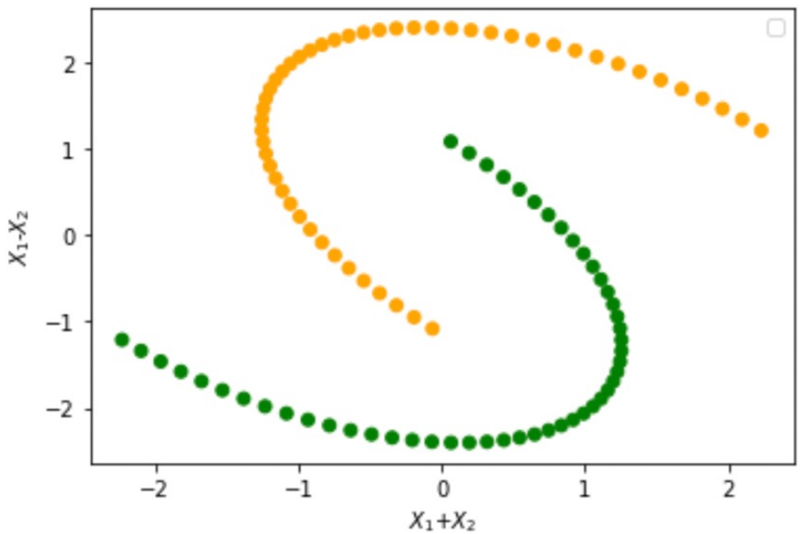
Moon dataset

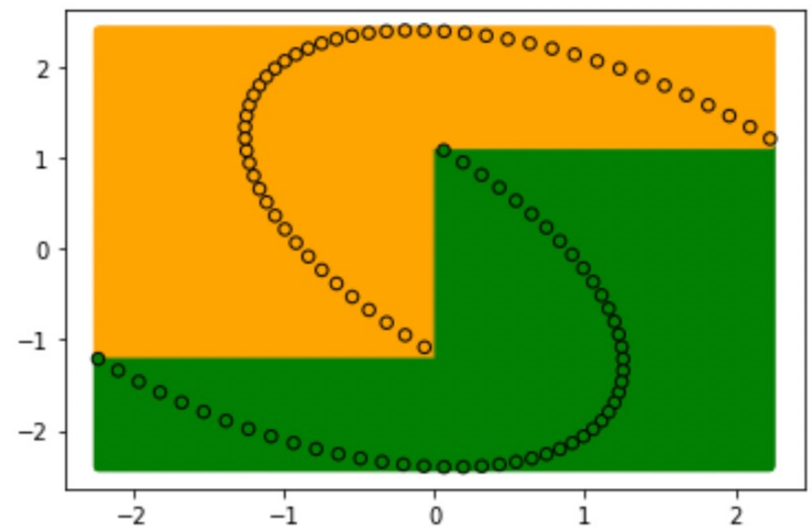




Benefits & Limitations

- Limitations:
  - Sensitive to orientation of training data





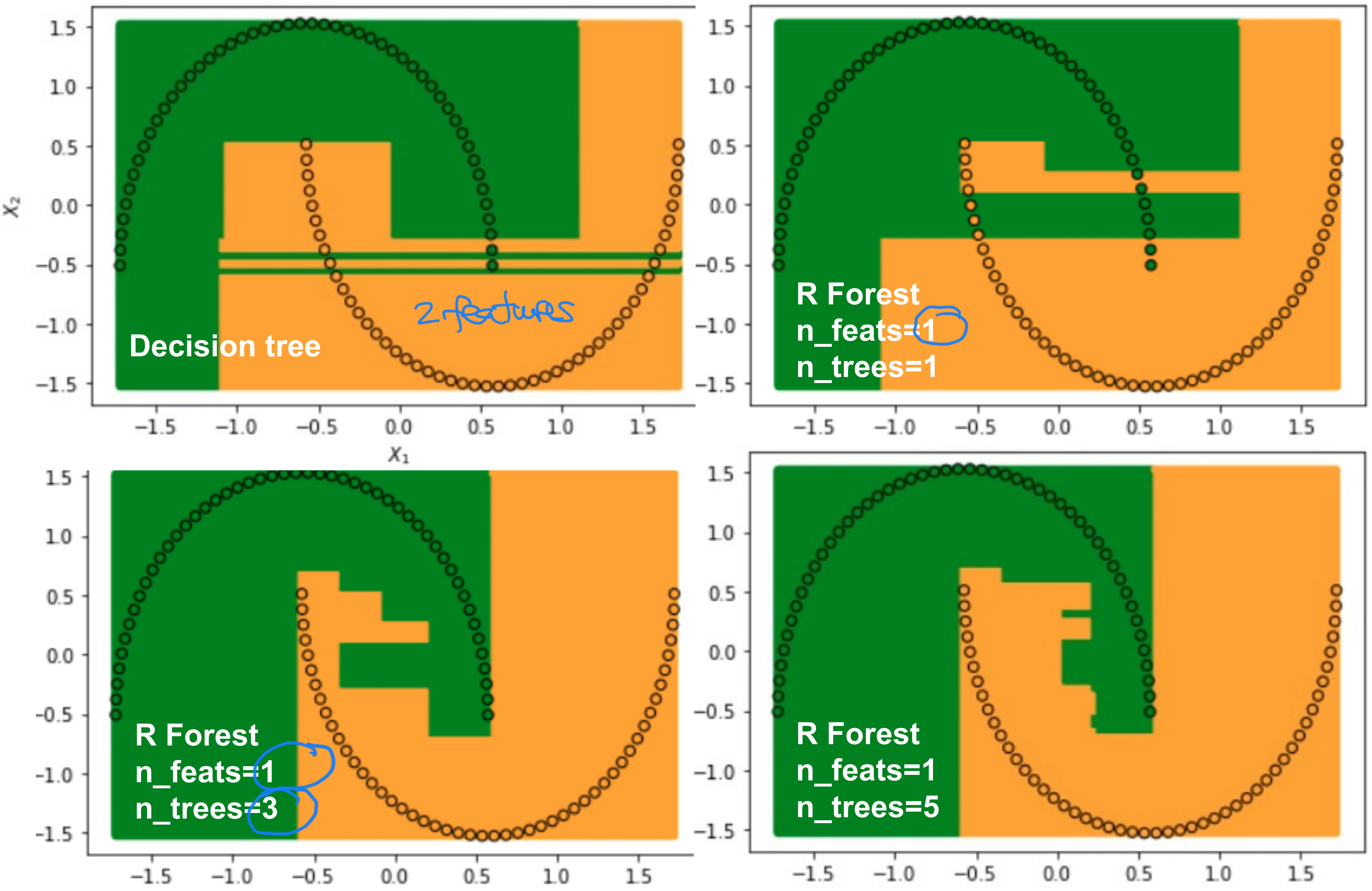
Benefits & Limitations

- Hyperparameters: Limit size of tree to reduce overfitting
  - Tree depth
  - Maximum no. of leaf nodes
  - Minimum no. of samples in each split
  - Minimum no. of samples in each leaf node
- Random forest: reduce model instability
  - averaging prediction over multiple decision trees
- PCA: identify the principal components

Random forests

- Uses Bagging (Bootstrap aggregation)
  - Average the predictions from a set of decision trees
  - Reduce variance (more stable prediction)
- For each tree: T_i D_i $T_1, \dots, T_{500}$ ← less correlated
  - Training set: randomly sample m examples with replacement from the original training set D
  - At each node: select $\tilde{n}$ features at random ($\tilde{n} \leq n$), seek best split on these features 3 10
- Prediction: stable prediction
  - Regression: average of predicted values over the trees
  - Classification: class with highest average predicted probability over the trees

Decision tree vs Random forest



Benefits & Limitations

- Reduce variance and prediction error
  - Training sets: bootstrap sampling
  - Feature bagging: reduce correlation among trees
  - Average prediction
- Feature importance
  - MDI: mean decrease in impurity due to a feature
  - Alternative measures: applicable to any model
  - Permutation importance (MDA): mean decrease in model performance if a feature's values on testing set are permuted

Ensemble methods

- Train a set of models (*base learners)*
- Prediction: from combining the predictions
- Bagging
  - For base learners with high variance, low bias
  - Average prediction of models trained independently
- Boosting
  - For base learners with low variance, high bias
  - train a sequence of models, one being dependent upon the previous models
  - eg. gradient boosting: train on the residual errors made by the previous model predictor
  - Reduce high bias of weak learners; hard to scale up