Intel Software

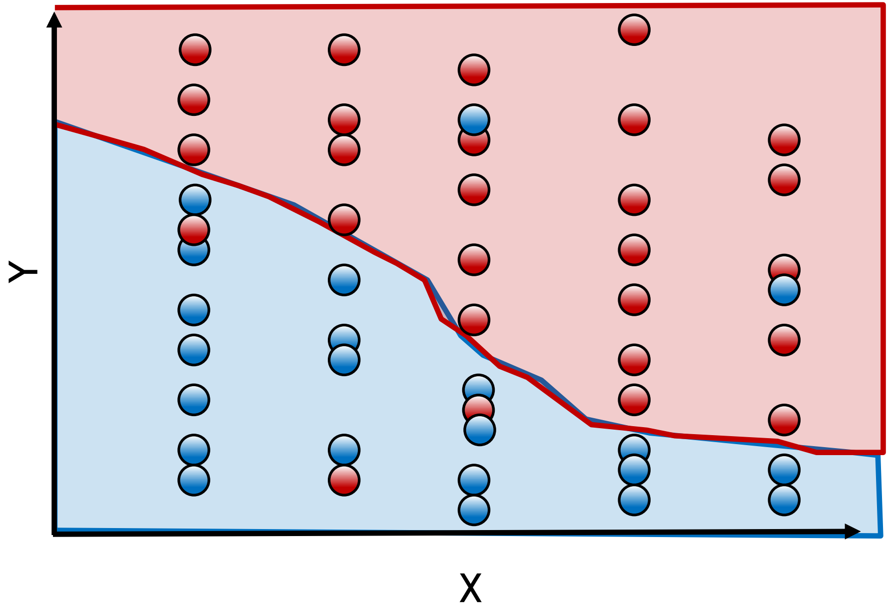
Decision Trees

Legal Notices and Disclaimers

Learning Objectives

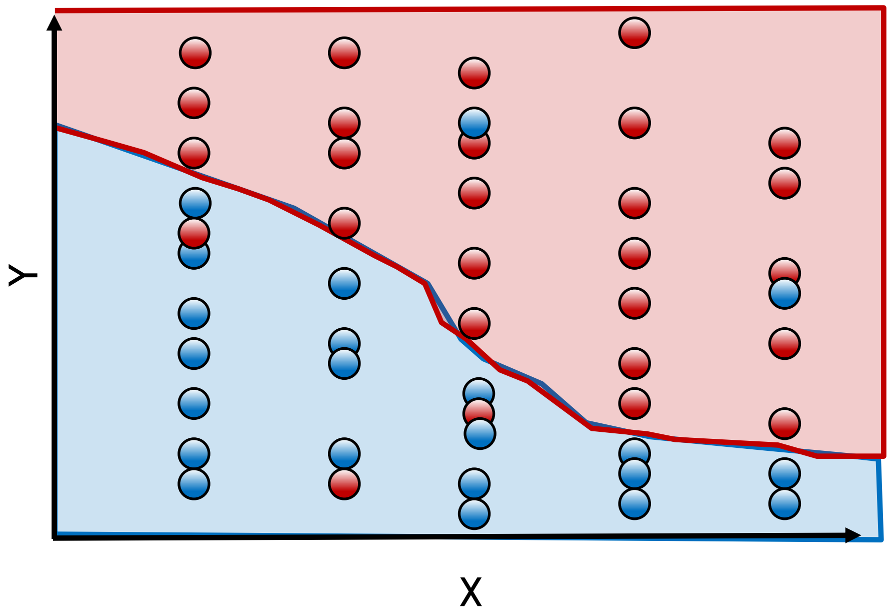
- Recognize how to use Decision trees for classification problems
- Consider more robust Scikit-learn* alternatives to Decision Trees
- Recognize how to identify the best split and the factors for splitting
- Explain strengths and weaknesses of decision trees
- Explain how regression trees help with classifying continuous values
- Apply Intel® Extension for Scikit-learn* to leverage underlying compute capabilities of hardware for Decision Tree alternatives

Overview of Classifier Characteristics



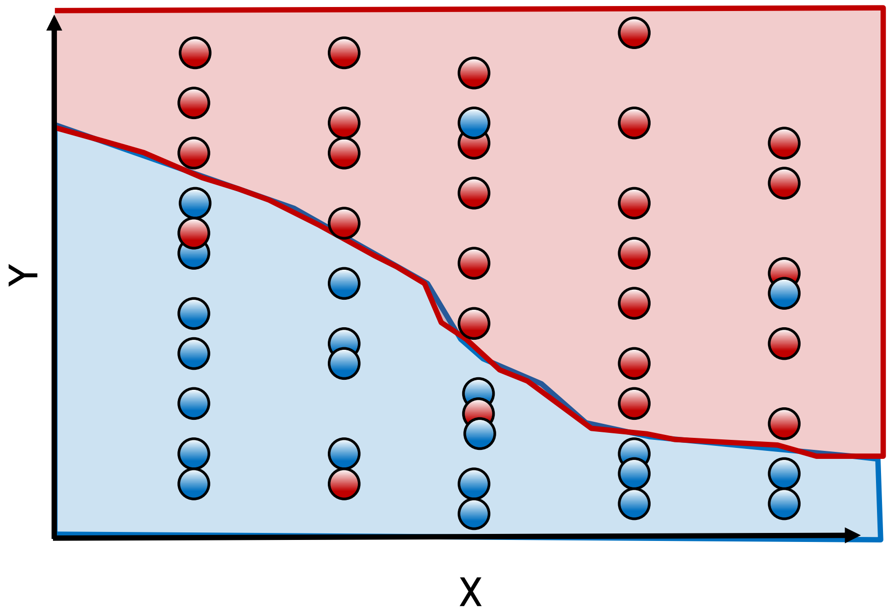
- For K-Nearest Neighbors, training data is the model

Overview of Classifier Characteristics



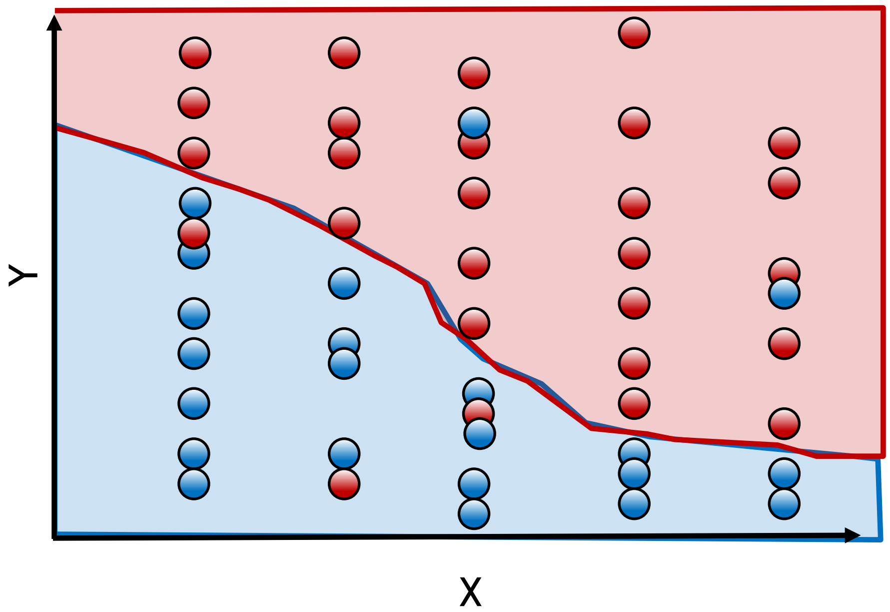
- For K-Nearest Neighbors, training data is the model
- Fitting is fast—just store data

Overview of Classifier Characteristics



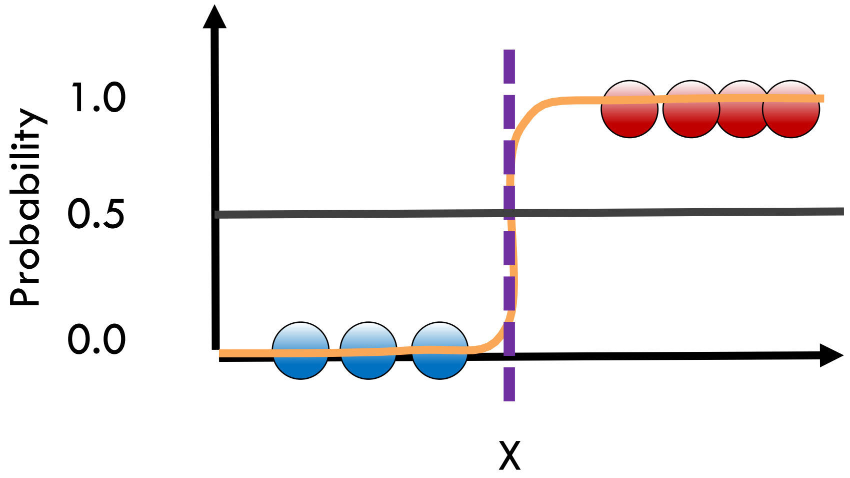
- For K-Nearest Neighbors, training data is the model
- Fitting is fast—just store data
- Prediction can be slow—lots of distances to measure

Overview of Classifier Characteristics



- For K-Nearest Neighbors, training data is the model
- Fitting is fast—just store data
- Prediction can be slow—lots of distances to measure
- Decision boundary is flexible

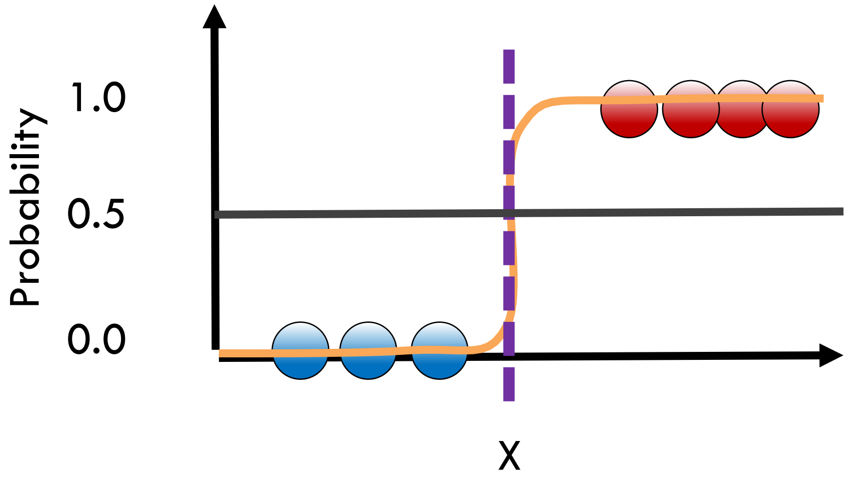
Overview of Classifier Characteristics



$$y_\beta(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x+\varepsilon)}}$$

- For logistic regression, model is just parameters

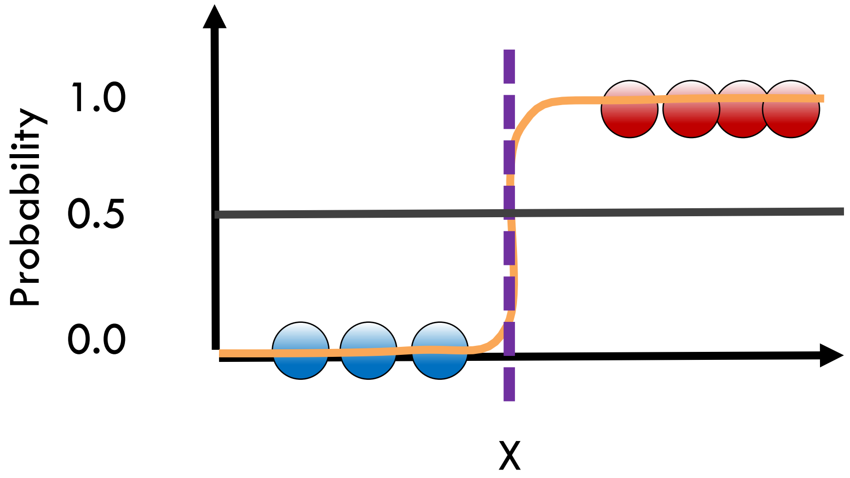
Overview of Classifier Characteristics



$$y_\beta(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x+\varepsilon)}}$$

- For logistic regression, model is just parameters
- Fitting can be slow—must find best parameters

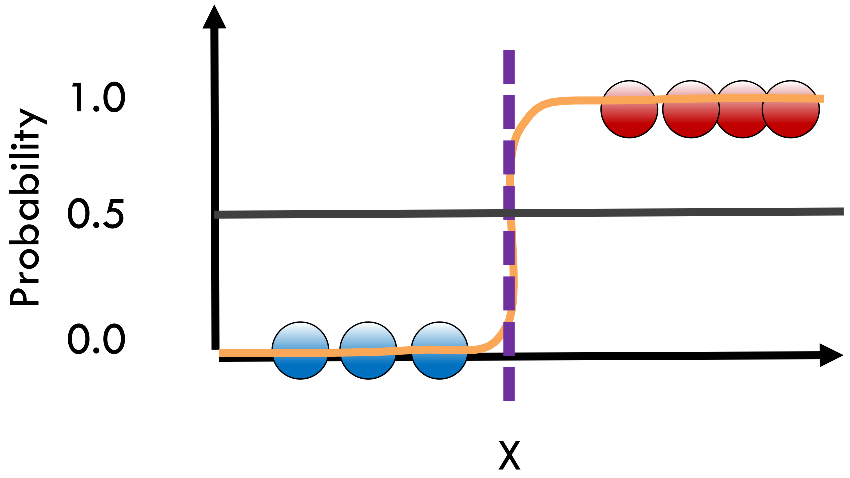
Overview of Classifier Characteristics



$$y_\beta(x) = \frac{1}{1+e^{-(\beta_0 + \beta_1 x + \varepsilon\ )}}$$

- For logistic regression, model is just parameters
- Fitting can be slow—must find best parameters
- Prediction is fast—calculate expected value

Overview of Classifier Characteristics



$$y_\beta(x) = \frac{1}{1+e^{-(\beta_0 + \beta_1 x + \varepsilon\ )}}$$

- For logistic regression, model is just parameters
- Fitting can be slow—must find best parameters
- Prediction is fast—calculate expected value
- Decision boundary is simple, less flexible

Introduction to Decision Trees

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Introduction to Decision Trees

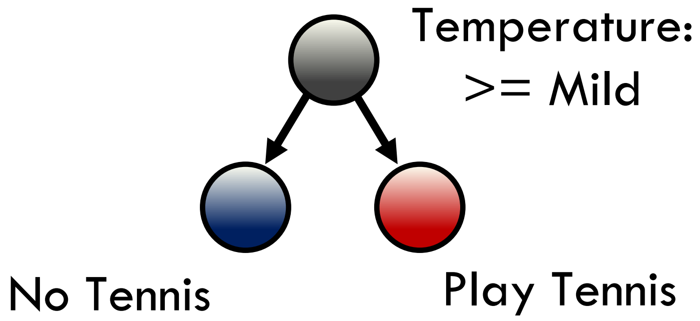
- Want to predict whether to play tennis based on temperature, humidity, wind, outlook

al

Introduction to Decision Trees

- Want to predict whether to play tennis based on temperature, humidity, wind, outlook
- Segment data based on features to predict result

al

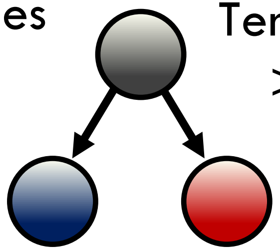


Introduction to Decision Trees

- Want to predict whether to play tennis based on temperature, humidity, wind, outlook

- Segment data based on features to predict result

al

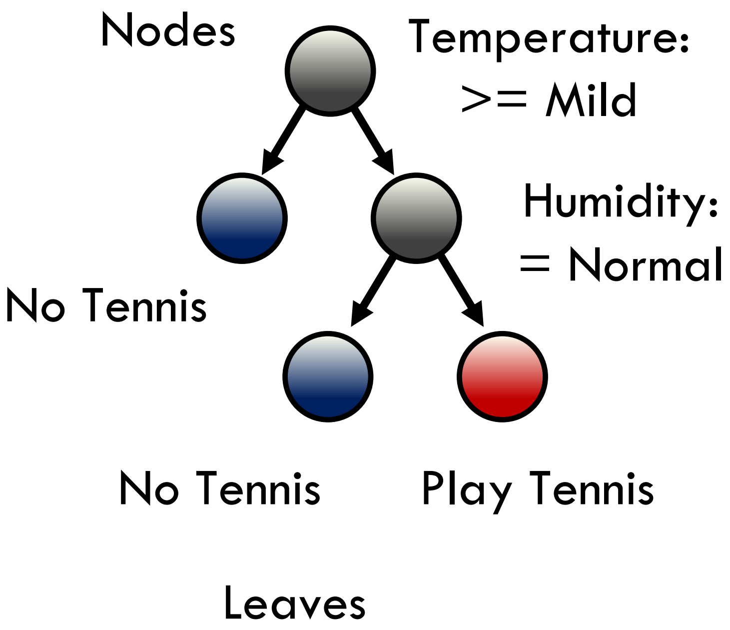


Introduction to Decision Trees

- Want to predict whether to play tennis based on temperature, humidity, wind, outlook

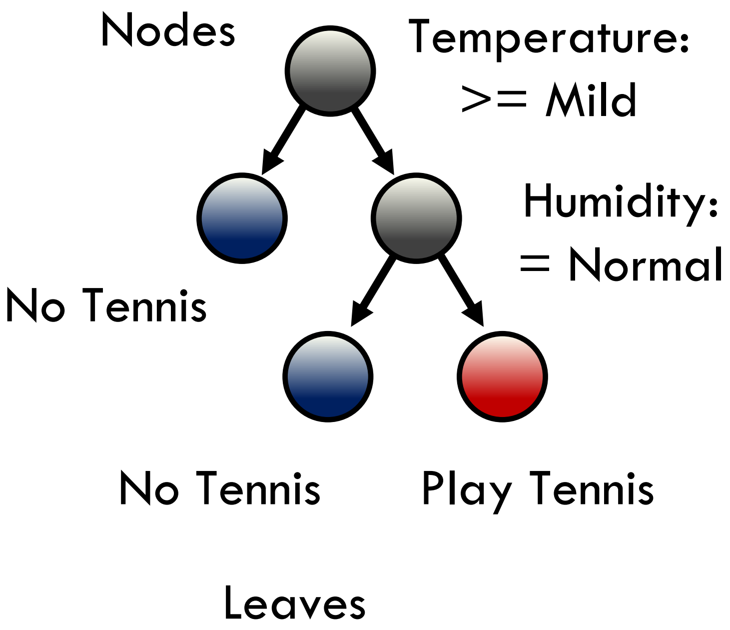
- Segment data based on features to predict result

al



Introduction to Decision Trees

- Want to predict whether to play tennis based on temperature, humidity, wind, outlook
- Segment data based on features to predict result
- Trees that predict categorical results are **decision trees**



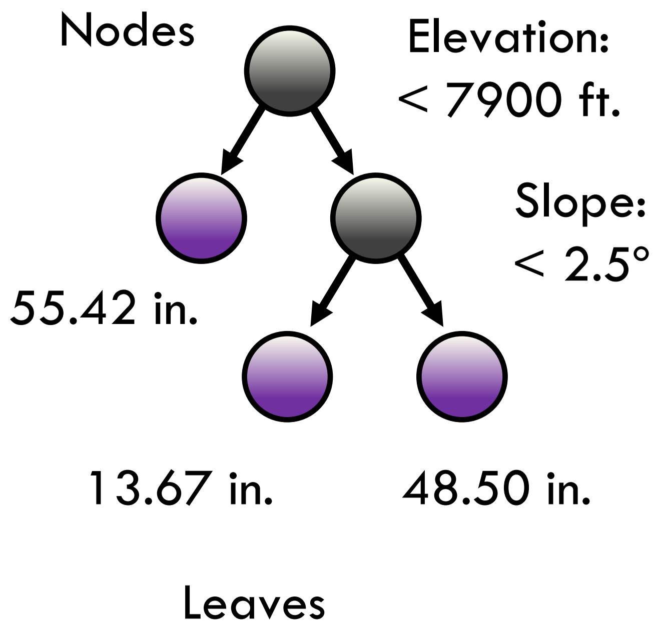
Regression Trees Predict Continuous Values

- Example: use slope an elevation in Himalayas

- Predict average precipitation (continuous value)

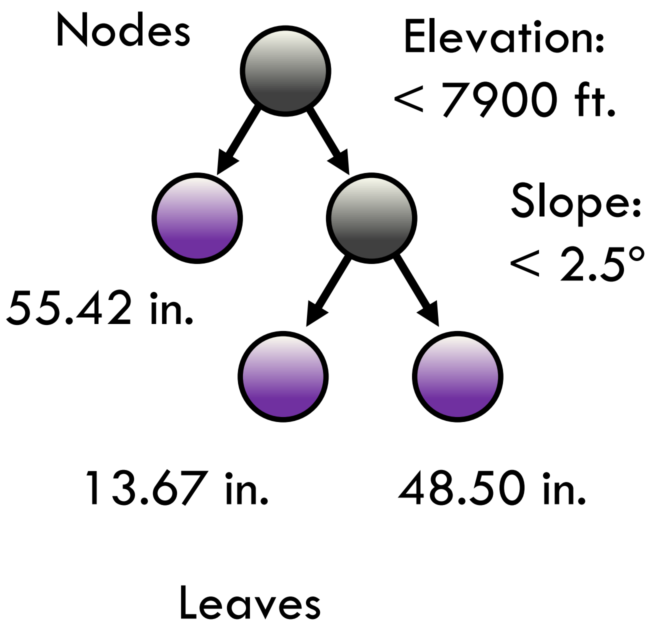
Regression Trees Predict Continuous Values

- Example: use slope an elevation in Himalayas
- Predict average precipitation (continuous value)

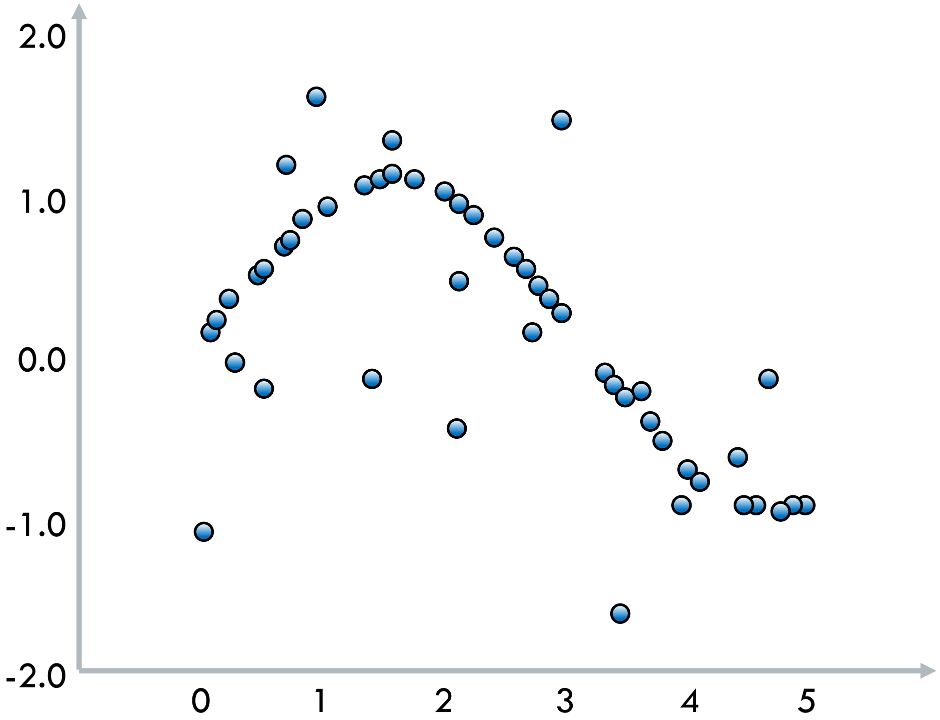


Regression Trees Predict Continuous Values

- Example: use slope an elevation in Himalayas
- Predict average precipitation (continuous value)
- Values at leaves are averages of members

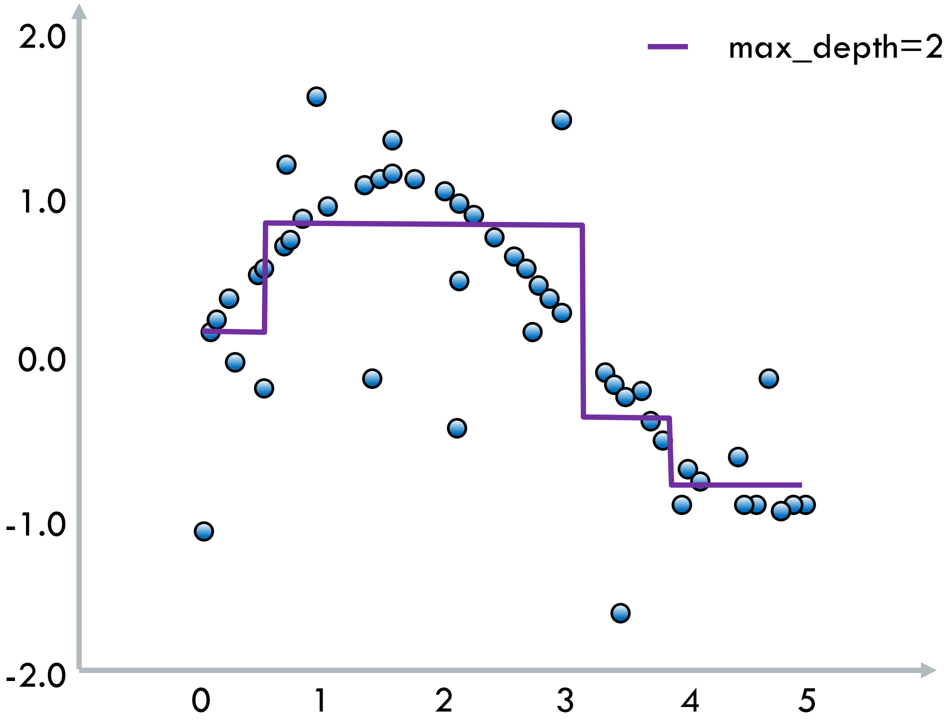


Regression Trees Predict Continuous Values



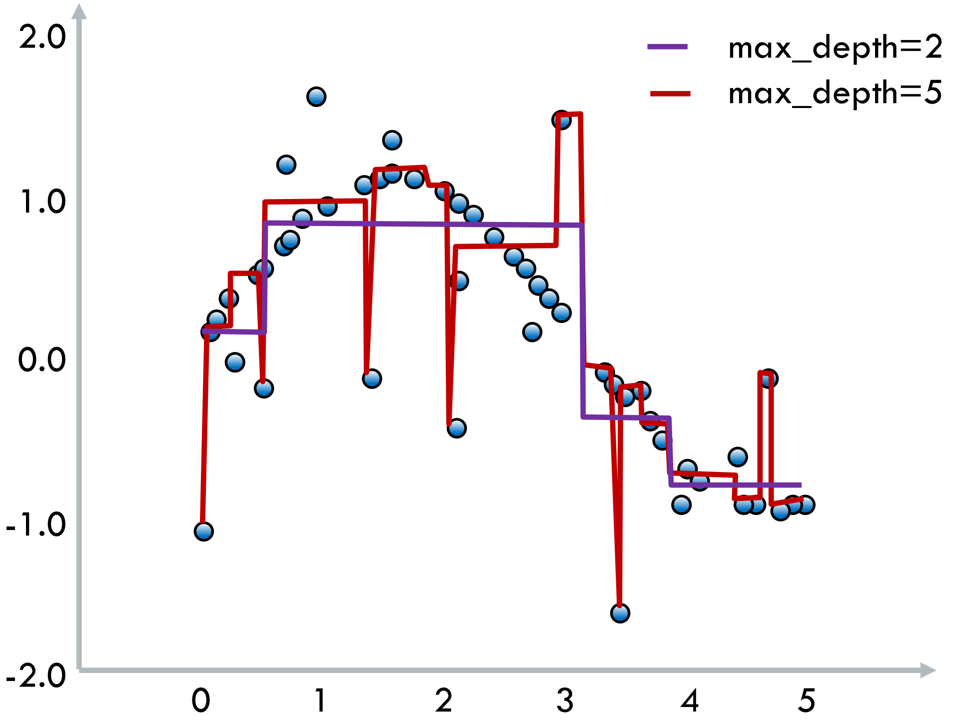
Source: http://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

Regression Trees Predict Continuous Values



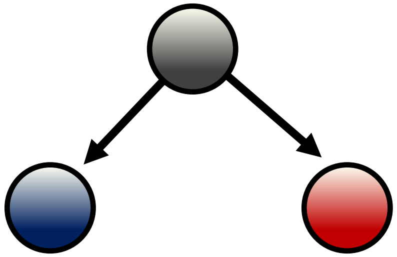
Source: http://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

Regression Trees Predict Continuous Values



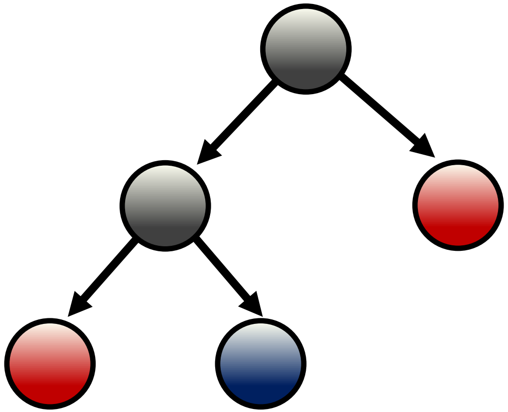
Source: http://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

Building a Decision Tree



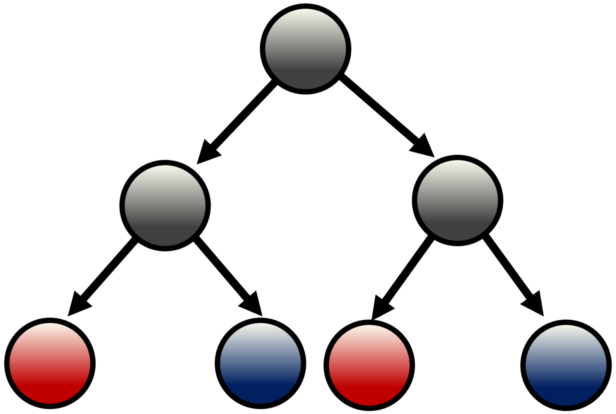
- Select a feature and split data into binary tree

Building a Decision Tree



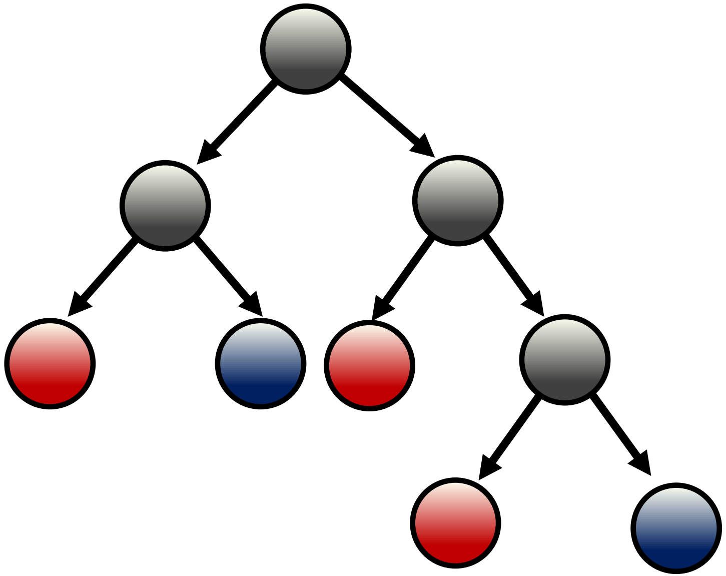
- Select a feature and split data into binary tree
- Continue splitting with available features

Building a Decision Tree



- Select a feature and split data into binary tree
- Continue splitting with available features

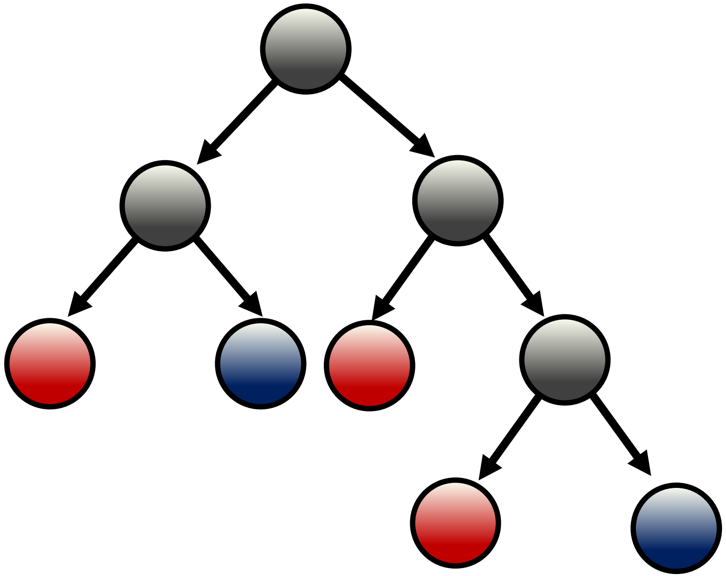
How Long to Keep Splitting?



ıe

Software

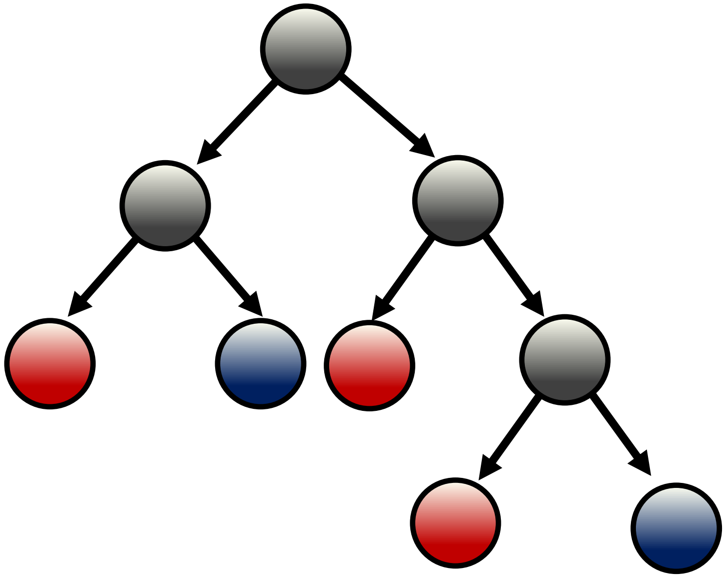
How Long to Keep Splitting?



Until:

- Leaf node(s) are pure (only one class remains)

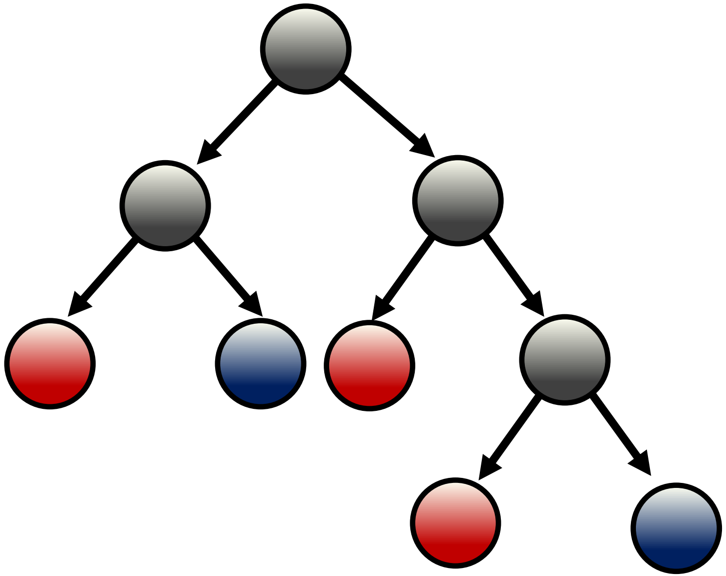
How Long to Keep Splitting?



Until:

- Leaf node(s) are pure (only one class remains)
- A maximum depth is reached

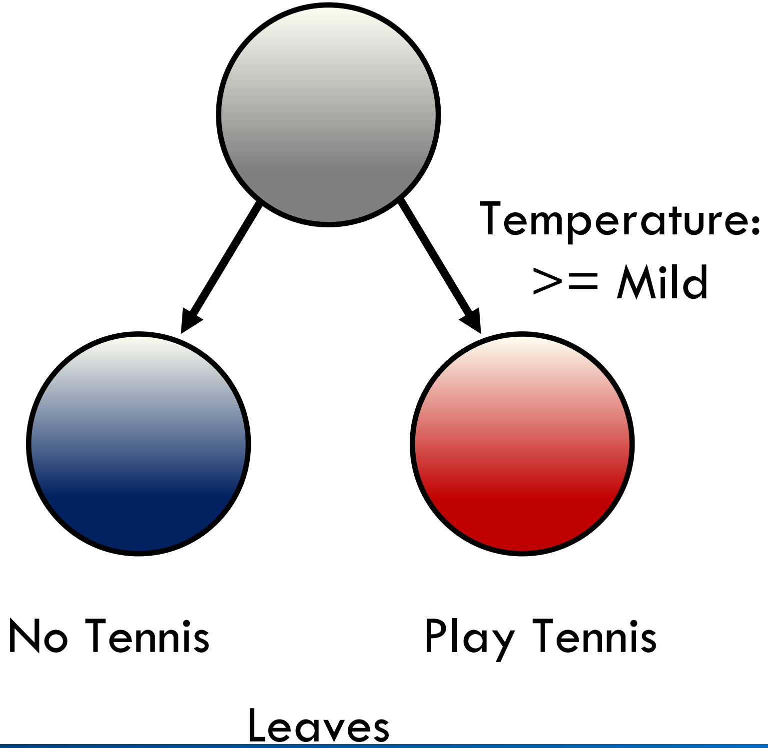
How Long to Keep Splitting?



Until:

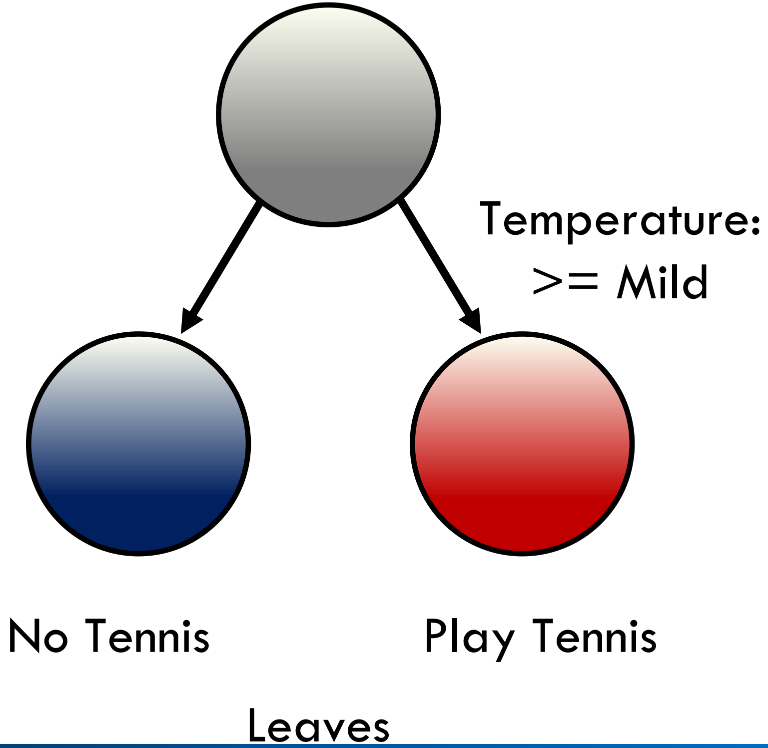
- Leaf node(s) are pure—only one class remains
- A maximum depth is reached
- A performance metric is achieved

Building the Best Decision Tree



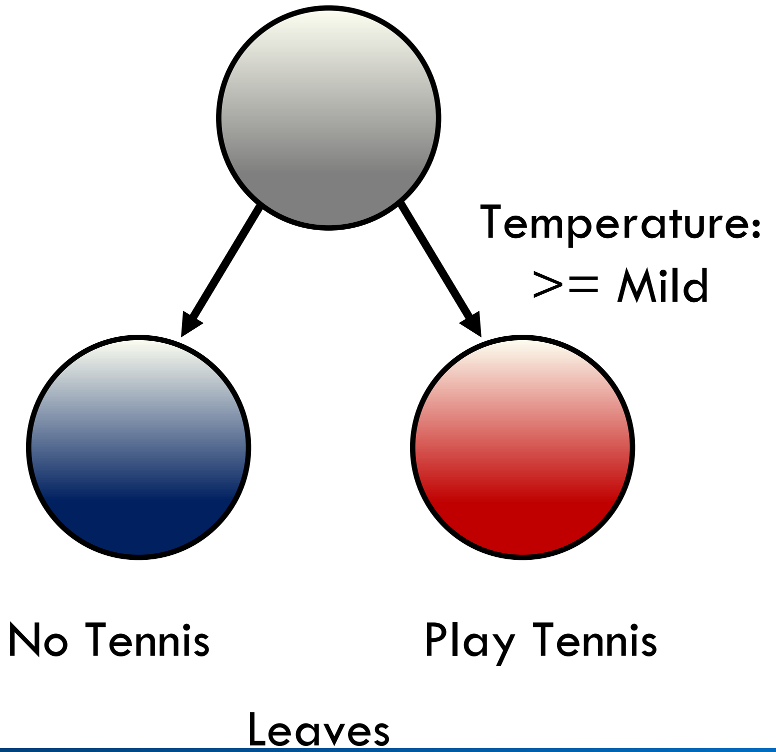
- Use greedy search: find the best split at each step

Building the Best Decision Tree



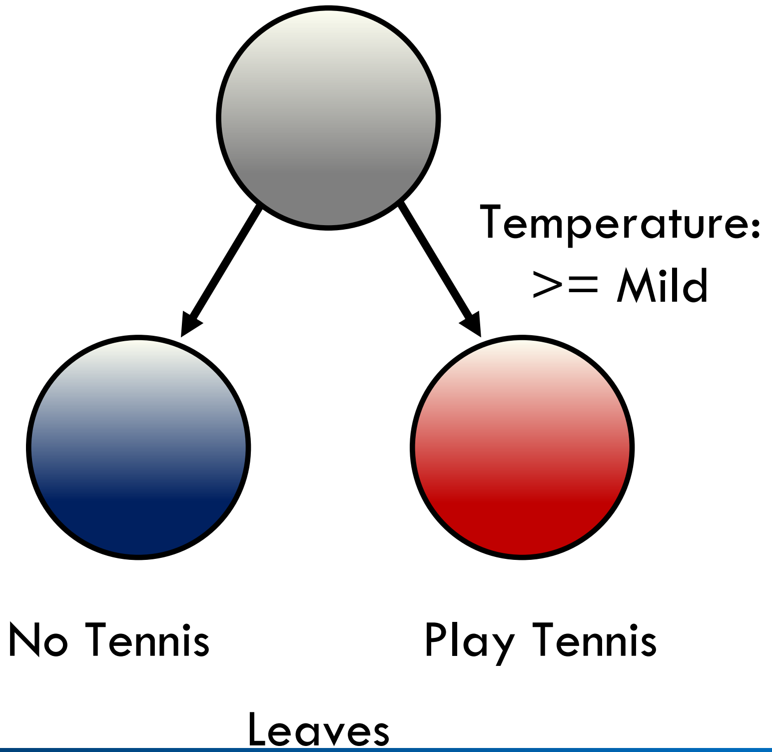
- Use greedy search: find the best split at each step
- What defines the best split?

Building the Best Decision Tree



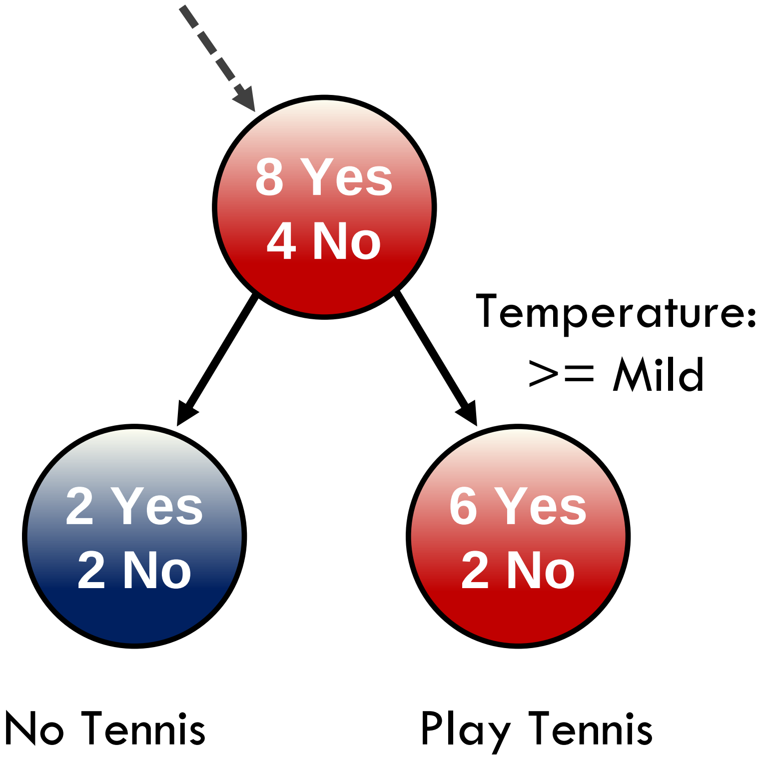
- Use greedy search: find the best split at each step
- What defines the best split?
- One that maximizes the information gained from the split

Building the Best Decision Tree



- Use greedy search: find the best split at each step
- What defines the best split?
- One that maximizes the information gained from the split
- How is information gain defined?

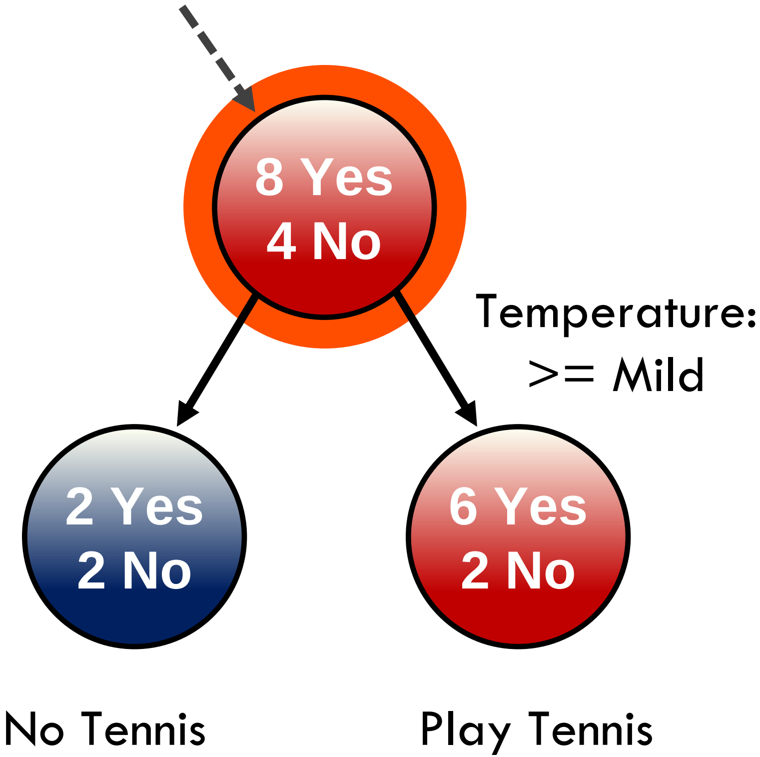
Splitting Based on Classification Error



Classification Error Equation

$$E(t) = 1 - \max_{i}[p(i|t)]$$

Splitting Based on Classification Error



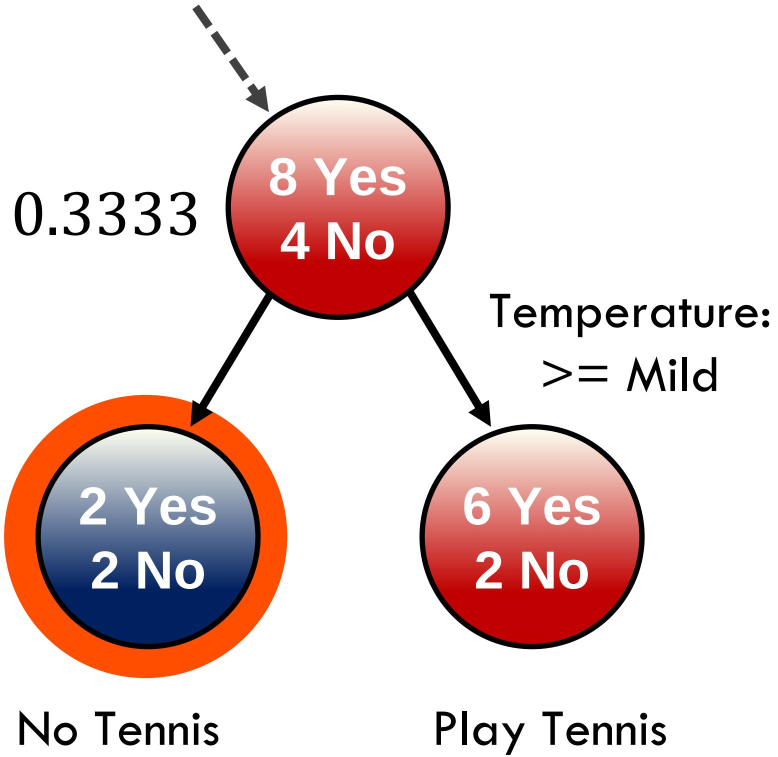
Classification Error Equation

$$E(t) = 1 - \max_i [p(i|t)]$$

Classification Error Before

$$1 - {}^{8}/_{12} = 0.3333$$

Splitting Based on Classification Error



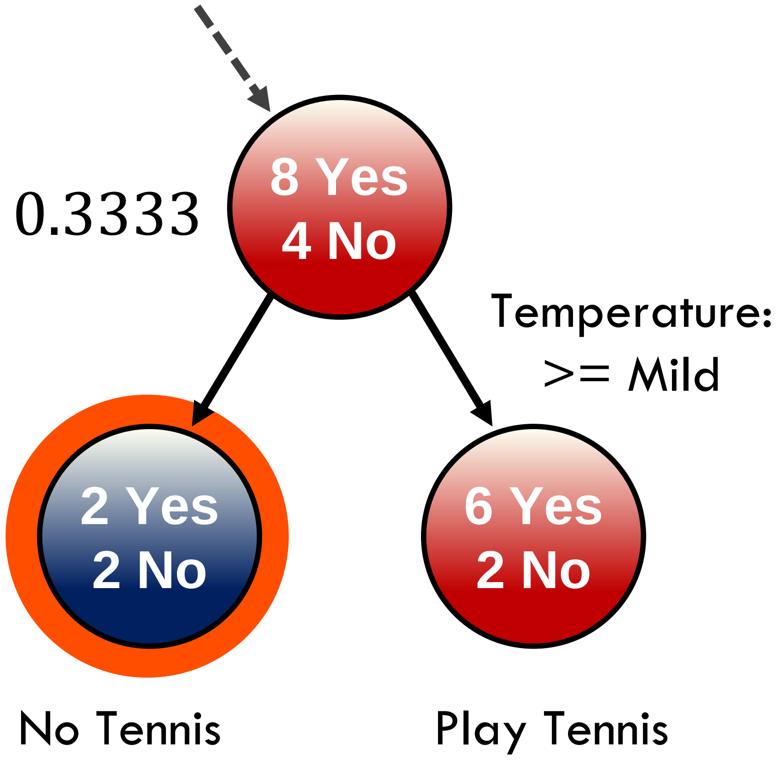
Classification Error Equation

$$E(t) = 1 - \max_i [p(i|t)]$$

Classification Error Left Side

$$1 - {}^{2}/_{4} = 0.5000$$

Splitting Based on Classification Error



Classification Error Equation

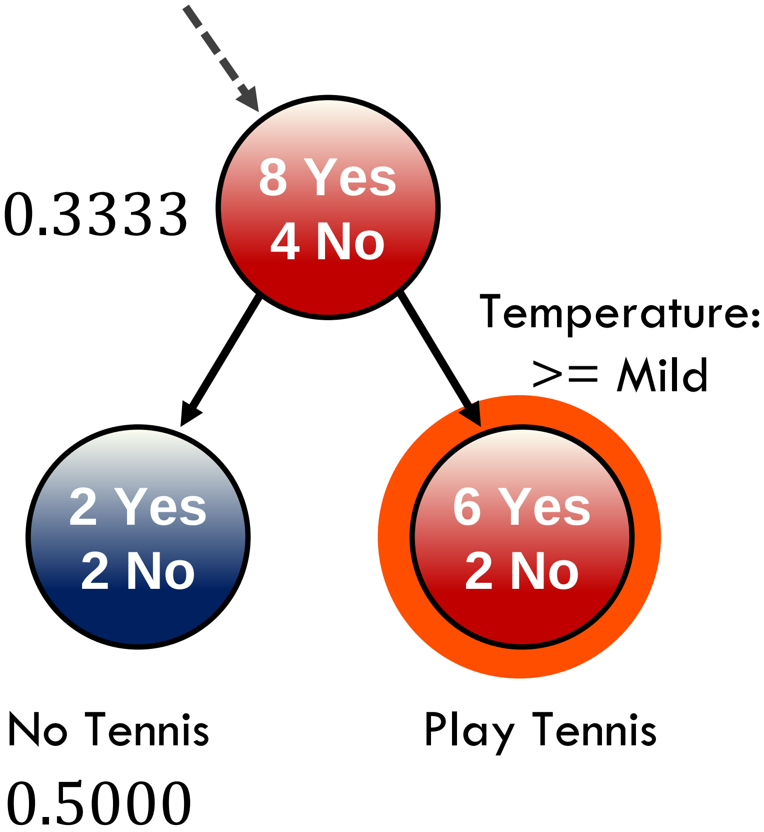
$$E(t) = 1 - \max_i [p(i|t)]$$

Classification Error Left Side

$$1 - {}^{2}/_{4} = 0.5000$$

Information lost on small # of data points

Splitting Based on Classification Error



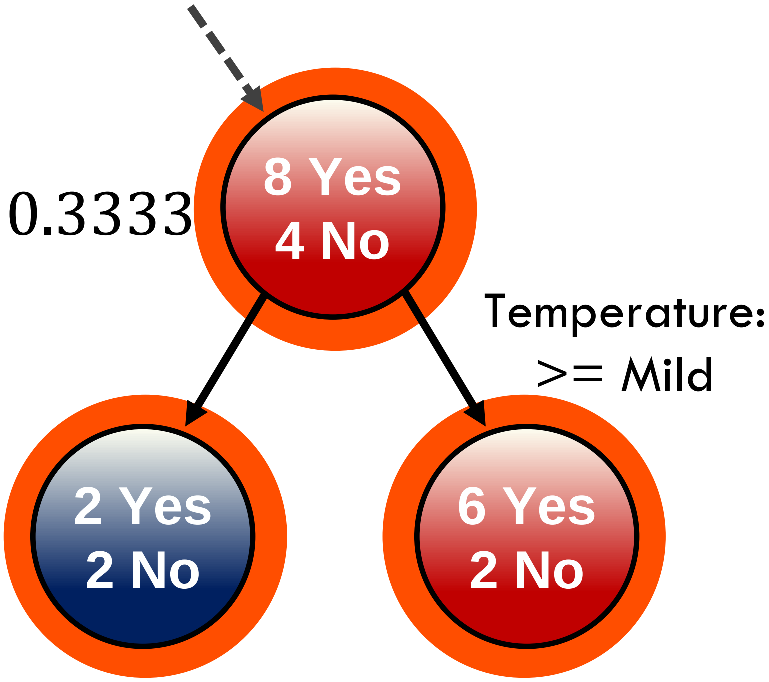
Classification Error Equation

$$E(t) = 1 - \max_i [p(i|t)]$$

Classification Error Right Side

$$1 - {}^{6}/_{8} = 0.2500$$

Splitting Based on Classification Error



No Tennis	Play Tennis
0.5000	0.2500

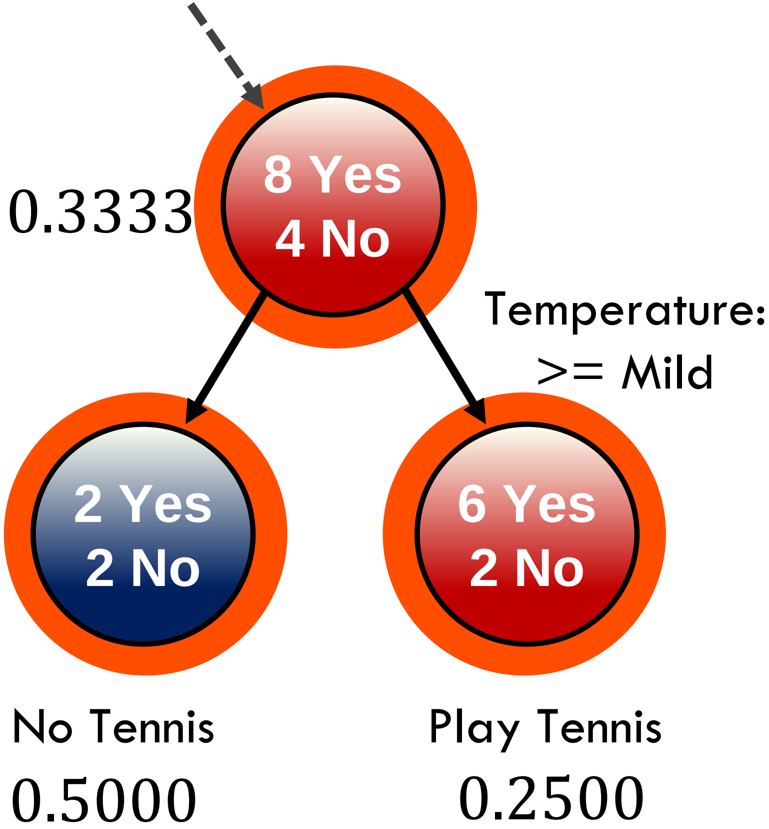
Classification Error Equation

$$E(t) = 1 - \max_i [p(i|t)]$$

Classification Error Change

$$0.3333 - {}^{4}/_{12} * 0.5000 - {}^{8}/_{12} * 0.2500$$

Splitting Based on Classification Error



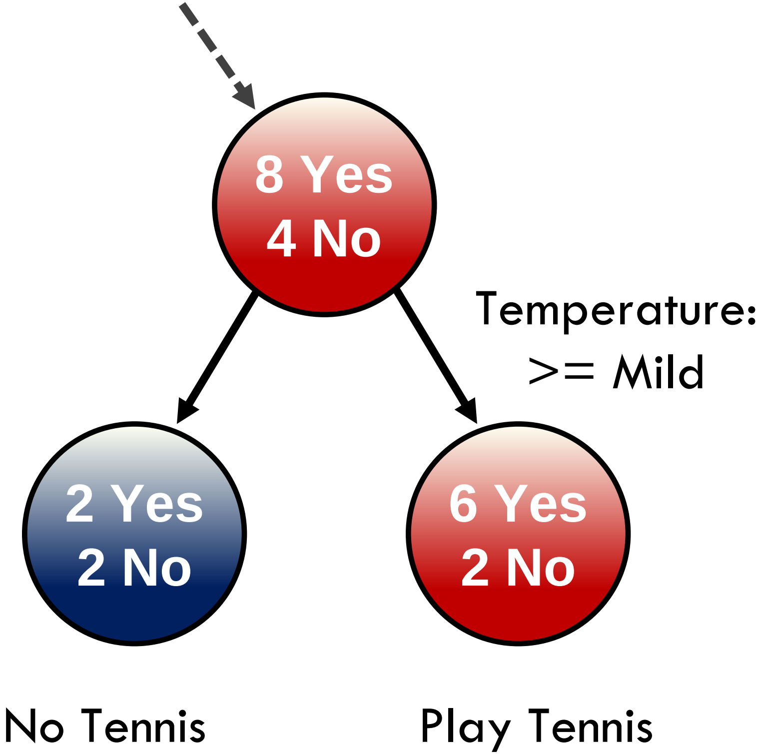
Classification Error Equation

$$E(t) = 1 - \max_i [p(i|t)]$$

Classification Error Change

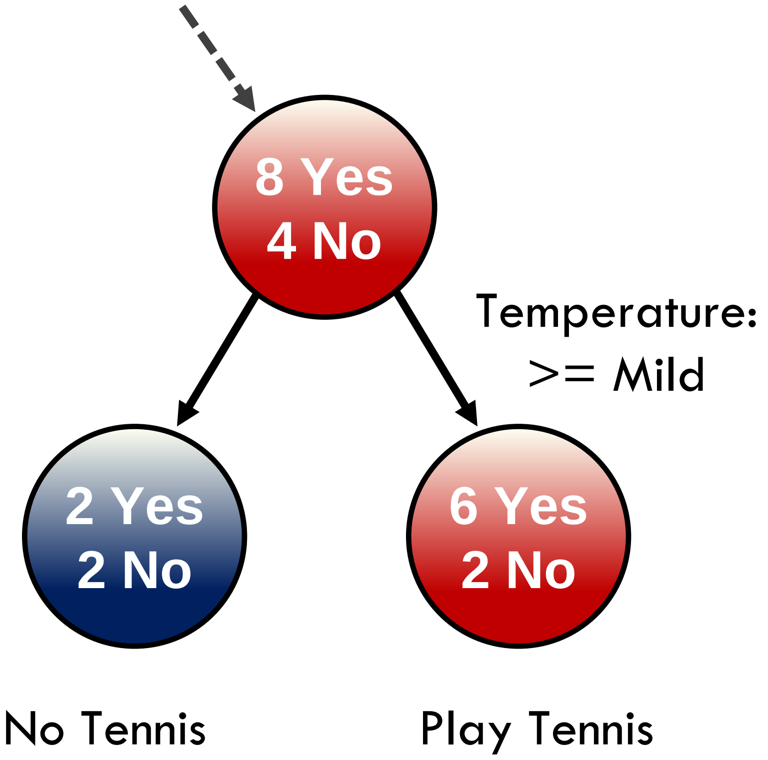
$$0.3333 \; - {}^{4}/_{12} * 0.5000 \; - {}^{8}/_{12} * 0.2500$$
$$= 0$$

Splitting Based on Classification Error



- Using classification error, no further splits would occur
- Problem: end nodes are not homogeneous
- Try a different metric?

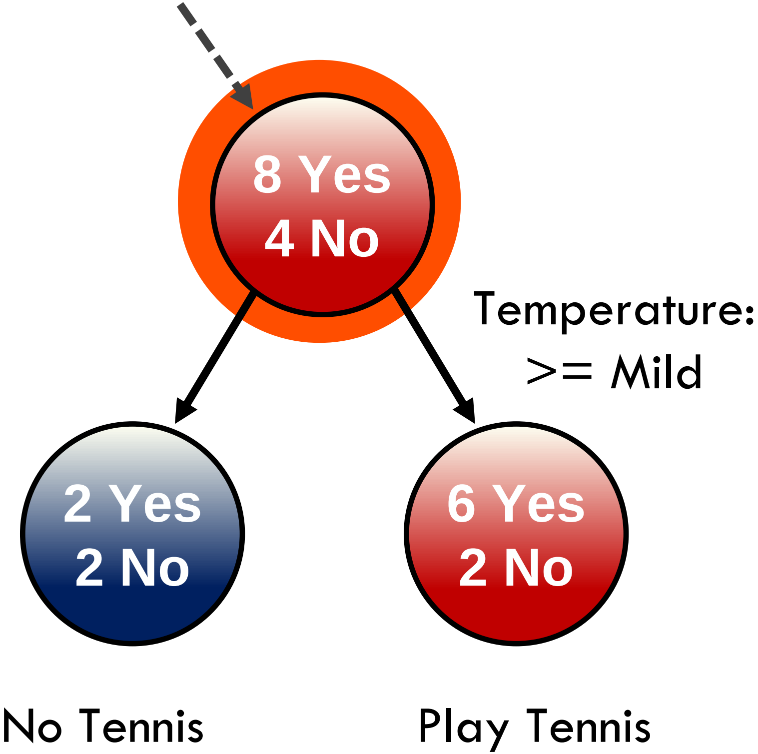
Splitting Based on Entropy



Entropy Equation

$$H(t) = -\sum_{i=1}^{n} p(i|t)log_2[p(i|t)]$$

Splitting Based on Entropy



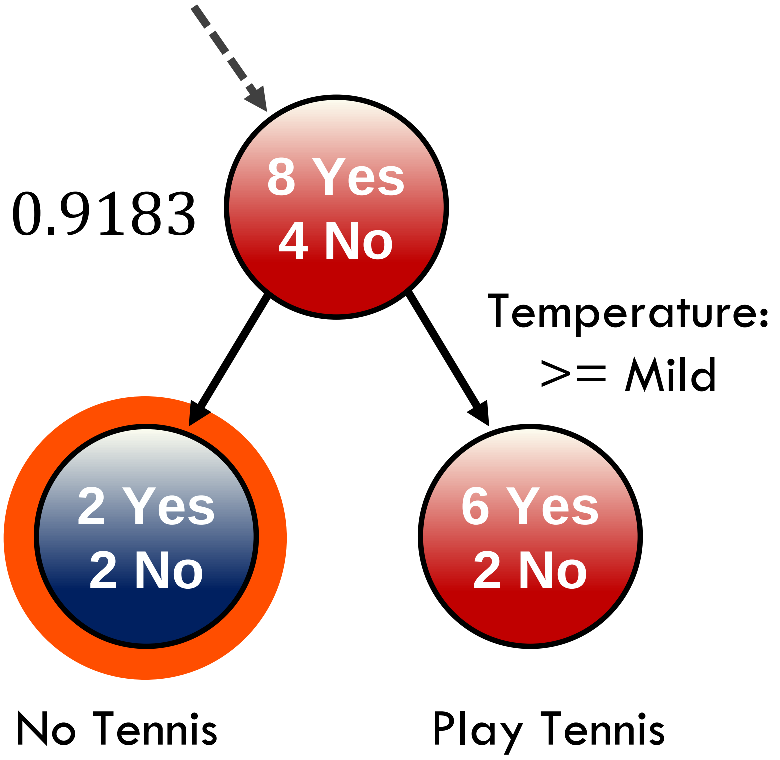
Entropy Equation

$$H(t) = -\sum_{i=1}^{n} p(i|t) log_2[p(i|t)]$$

Entropy Before

$$-{}^{8}/_{12}\, log_2({}^{8}/_{12}) \; - {}^{4}/_{12}\, log_2({}^{4}/_{12}) = 0.9183$$

Splitting Based on Entropy



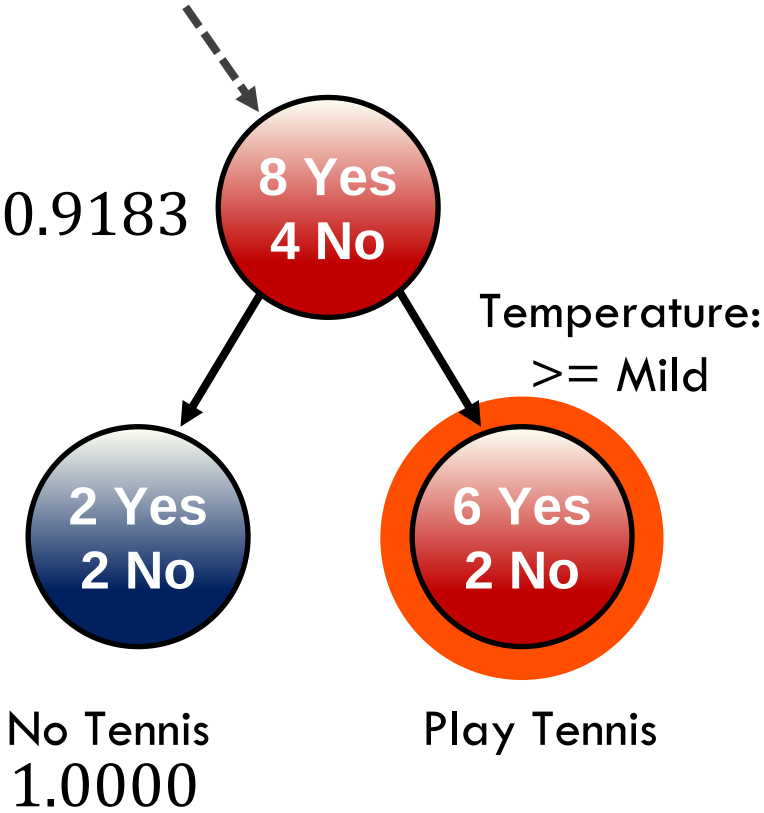
Entropy Equation

$$H(t) = -\sum_{i=1}^{n} p(i|t) log_2[p(i|t)]$$

Entropy Left Side

$$-{}^{2}/_{4}\, log_2({}^{2}/_{4}) \; - {}^{2}/_{4}\, log_2({}^{2}/_{4}) = 1.0000$$

Splitting Based on Entropy



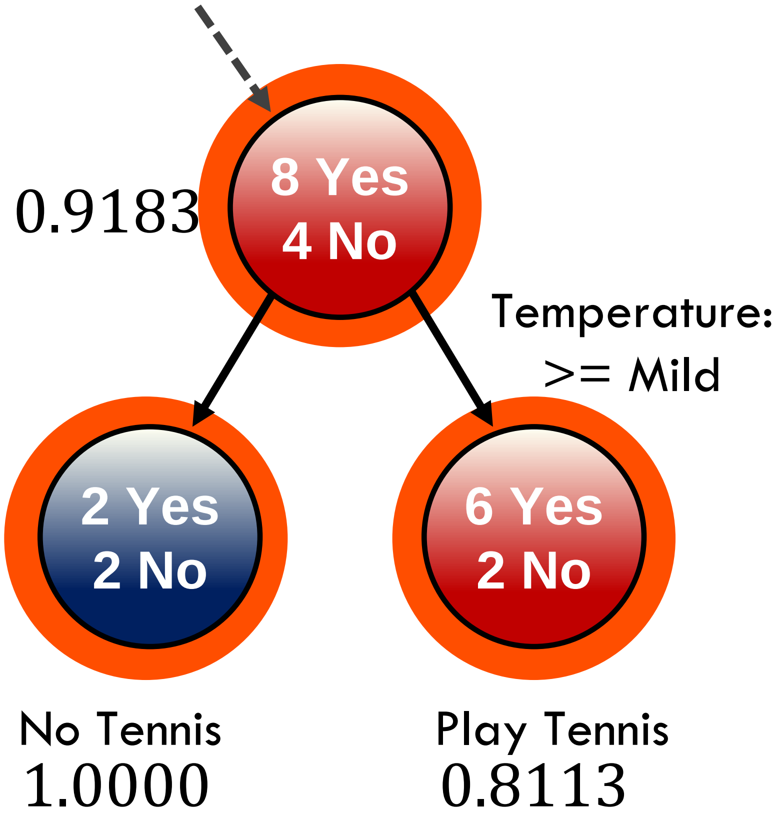
Entropy Equation

$$H(t) = -\sum_{i=1}^{n} p(i|t) log_2[p(i|t)]$$

Entropy Right Side

$$-{}^{6}/_{8}\, log_2({}^{6}/_{8}) \; -{}^{2}/_{8}\, log_2({}^{2}/_{8}) = 0.8113$$

Splitting Based on Entropy



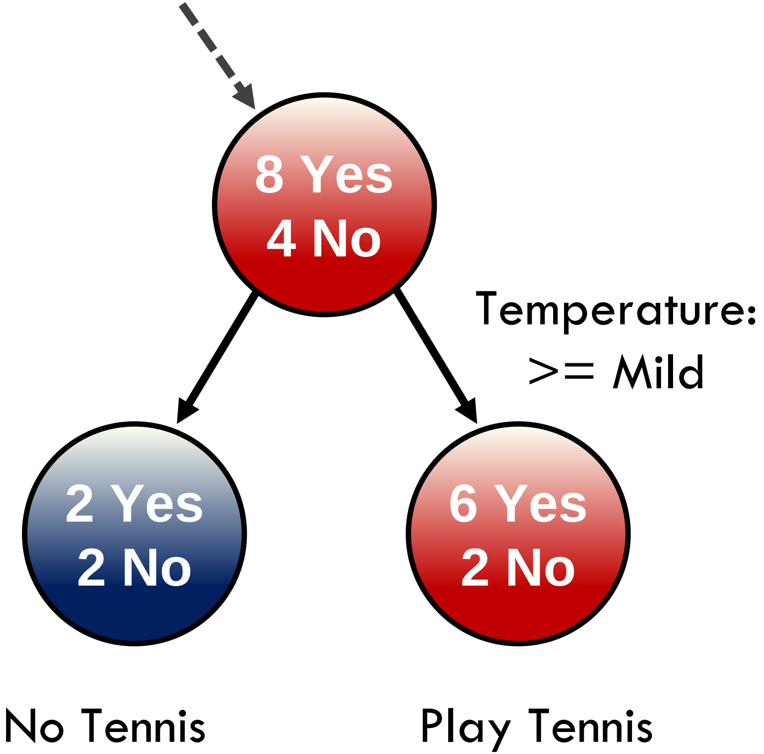
Entropy Equation

$$H(t) = -\sum_{i=1}^{n} p(i|t) log_2[p(i|t)]$$

Entropy Change

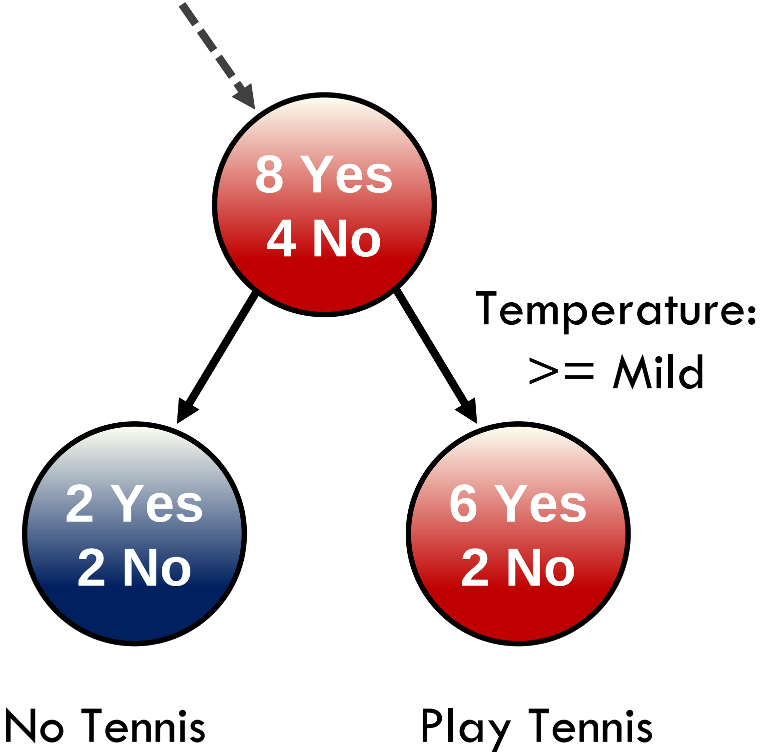
$$0.9183 - {}^{4}/_{12} * 1.0000 - {}^{8}/_{12} * 0.8113$$
$$= 0.0441$$

Splitting Based on Entropy



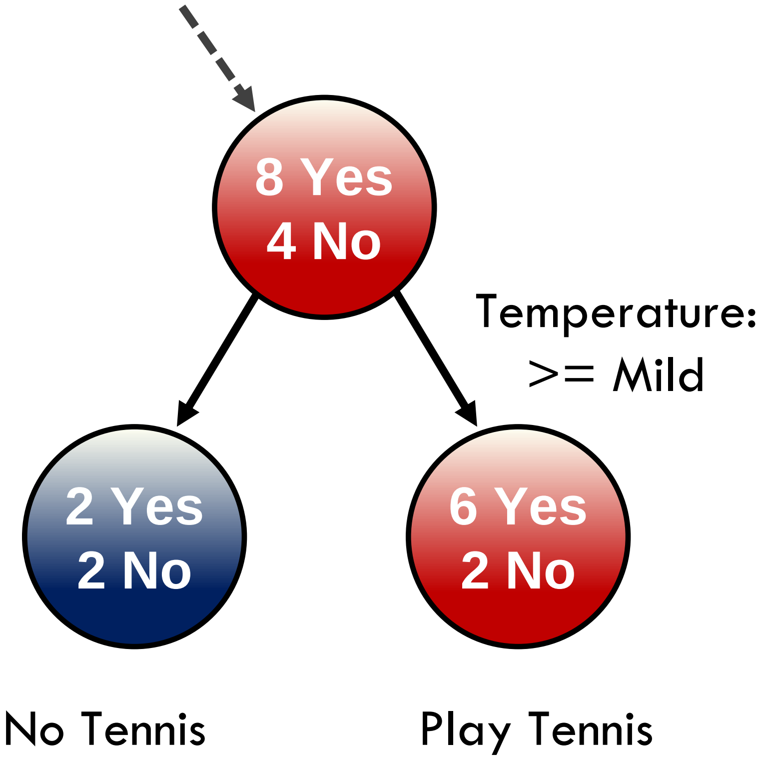
- Splitting based on entropy allows further splits to occur

Splitting Based on Entropy



- Splitting based on entropy allows further splits to occur
- Can eventually reach goal of homogeneous nodes

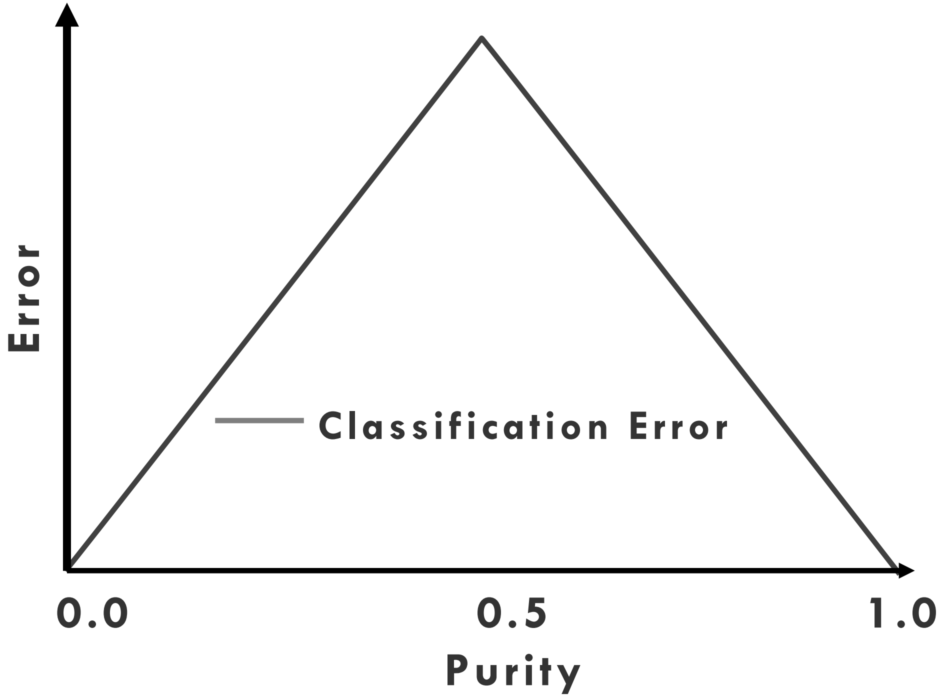
Splitting Based on Entropy



- Splitting based on entropy allows further splits to occur
- Can eventually reach goal of homogeneous nodes
- Why does this work with entropy but not classification error?

Classification Error vs Entropy

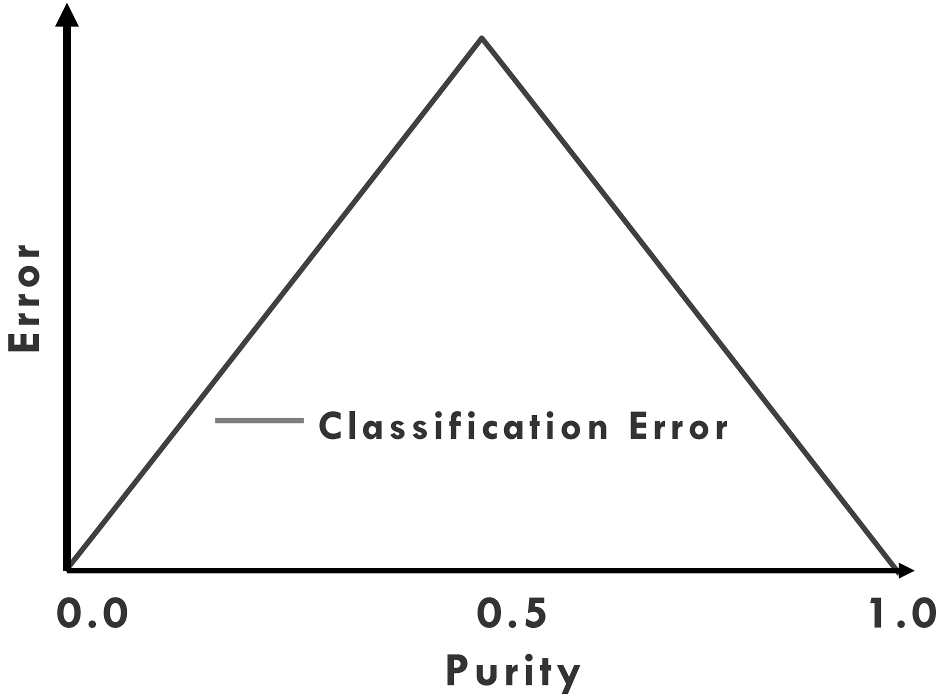
- Classification error is a flat function with maximum at center
- Center represents ambiguity—



$$E(t) = 1 - \max_i [p(i|t)]$$

Classification Error vs Entropy

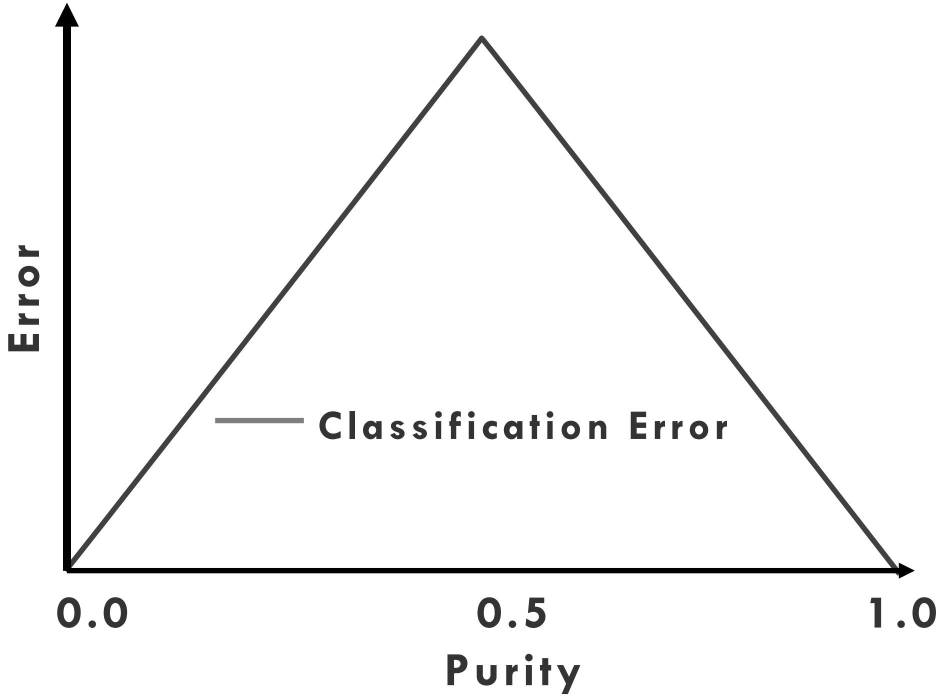
- Classification error is a flat function with maximum at center
- Center represents ambiguity—50/50 split
- Splitting metrics favor results that



$$E(t) = 1 - \max_i [p(i|t)]$$

Classification Error vs Entropy

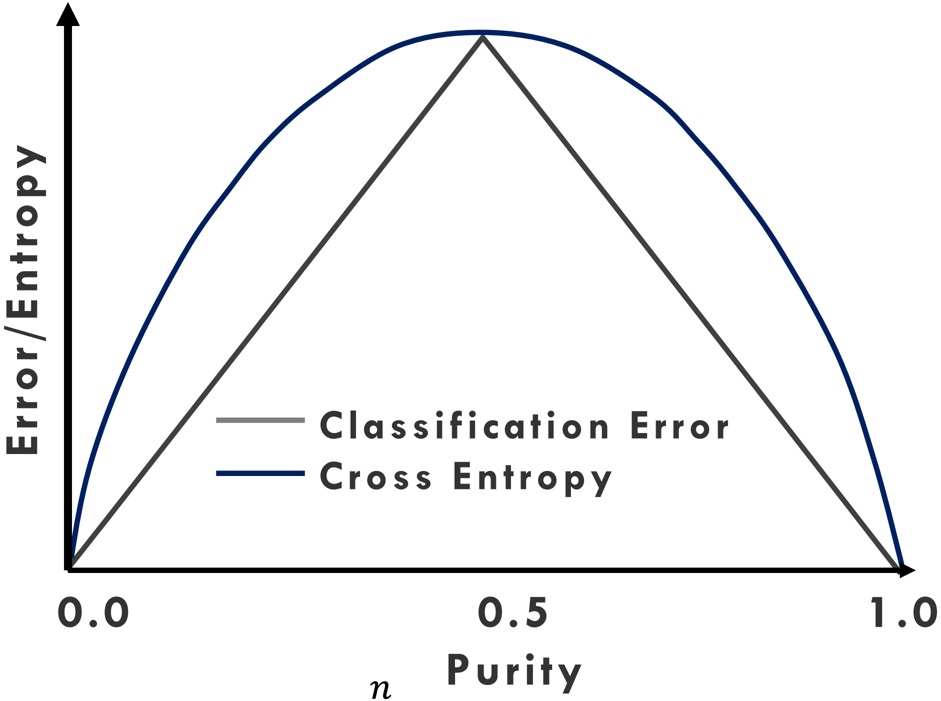
- Classification error is a flat function with maximum at center
- Center represents ambiguity—50/50 split
- Splitting metrics favor results that are furthest away from the center



$$E(t) = 1 - \max_i [p(i|t)]$$

Classification Error vs Entropy

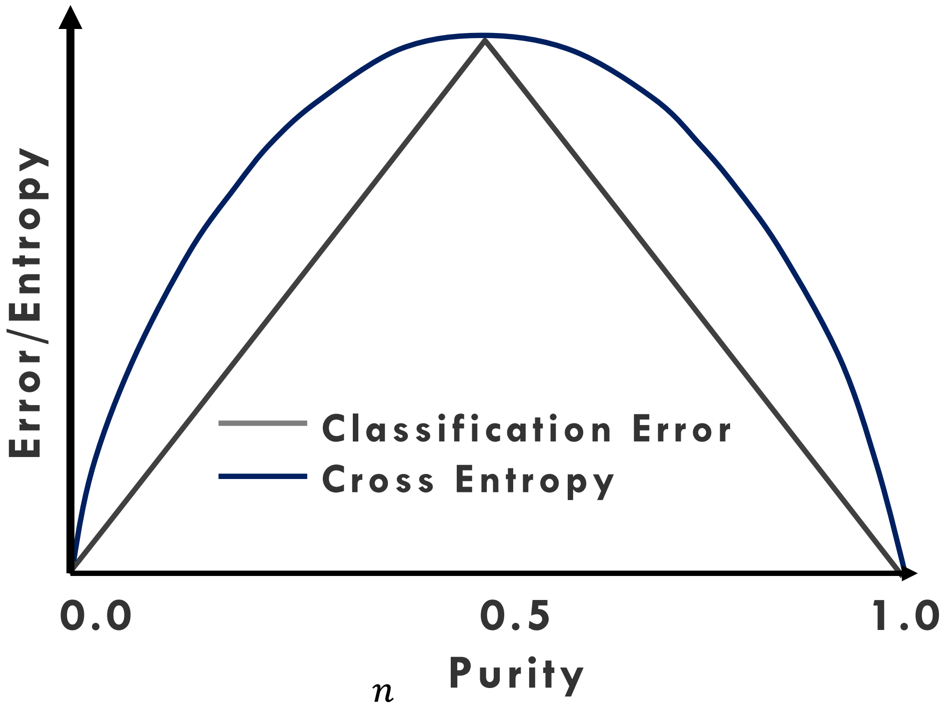
- Entropy has the same maximum but is curved



$$H(t) = -\sum_{i=1}^{n} p(i|t) log_2[p(i|t)]$$

Classification Error vs Entropy

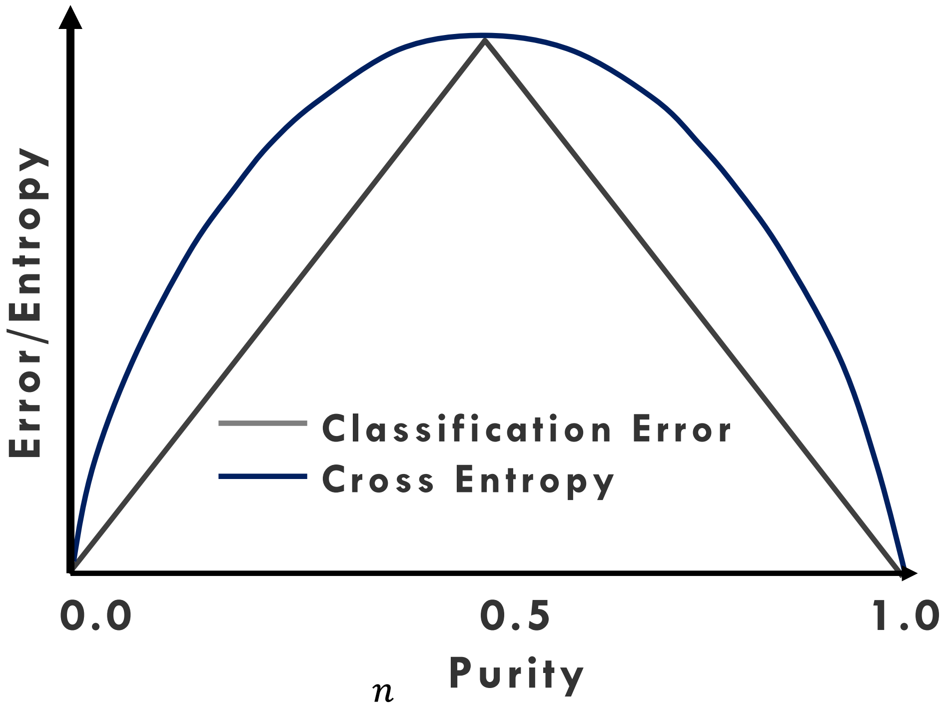
- Entropy has the same maximum but is curved
- Curvature allows splitting to continue until nodes are pure



$$H(t) = -\sum_{i=1}^{n} p(i|t) log_2[p(i|t)]$$

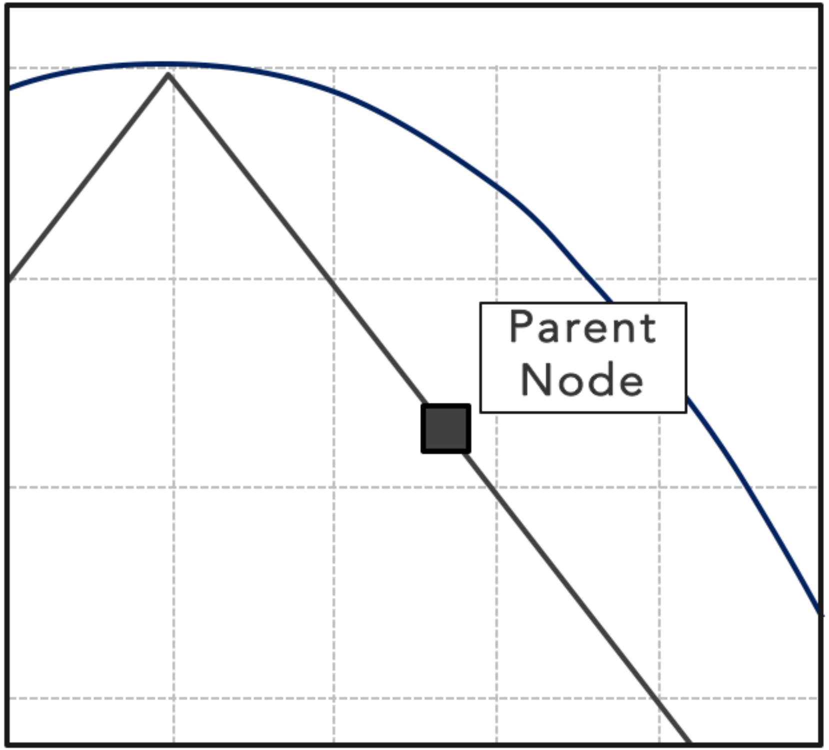
Classification Error vs Entropy

- Entropy has the same maximum but is curved
- Curvature allows splitting to continue until nodes are pure
- How does this work?



$$H(t) = -\sum_{i=1}^{n} p(i|t) log_2[p(i|t)]$$

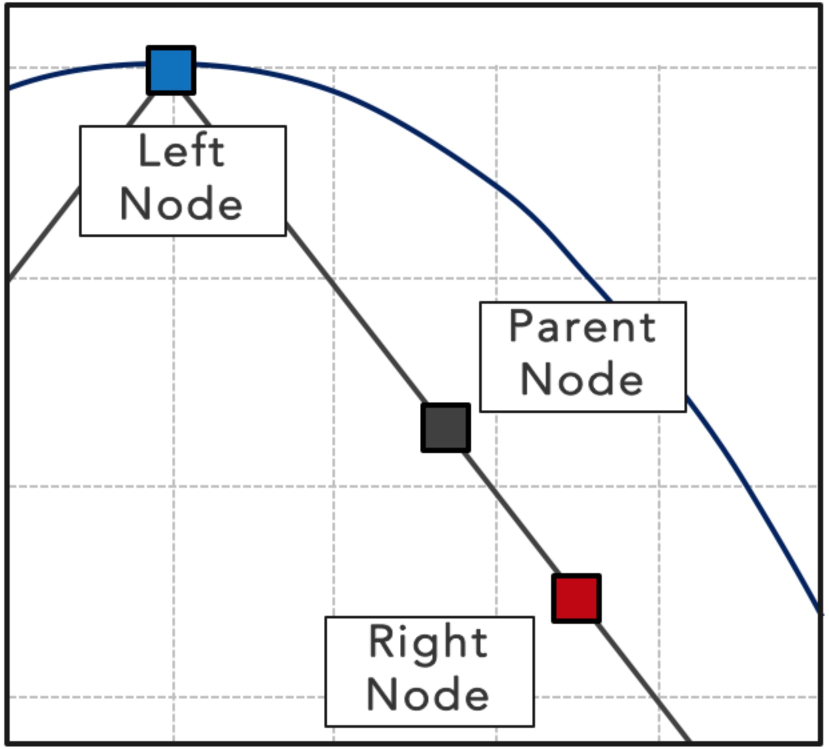
Information Gained by Splitting



- With classification error, the function is flat

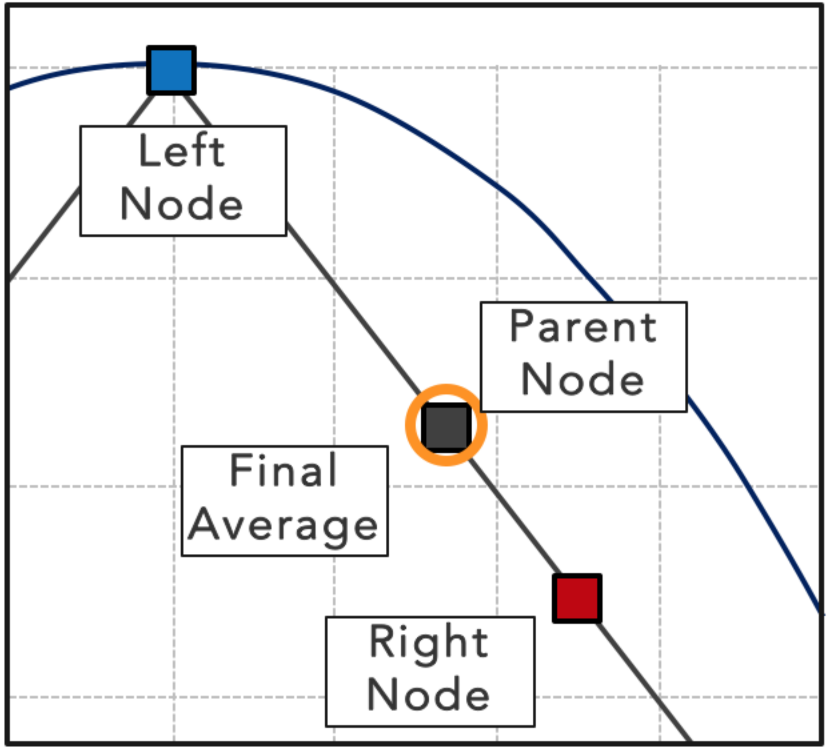
ɔr

Information Gained by Splitting



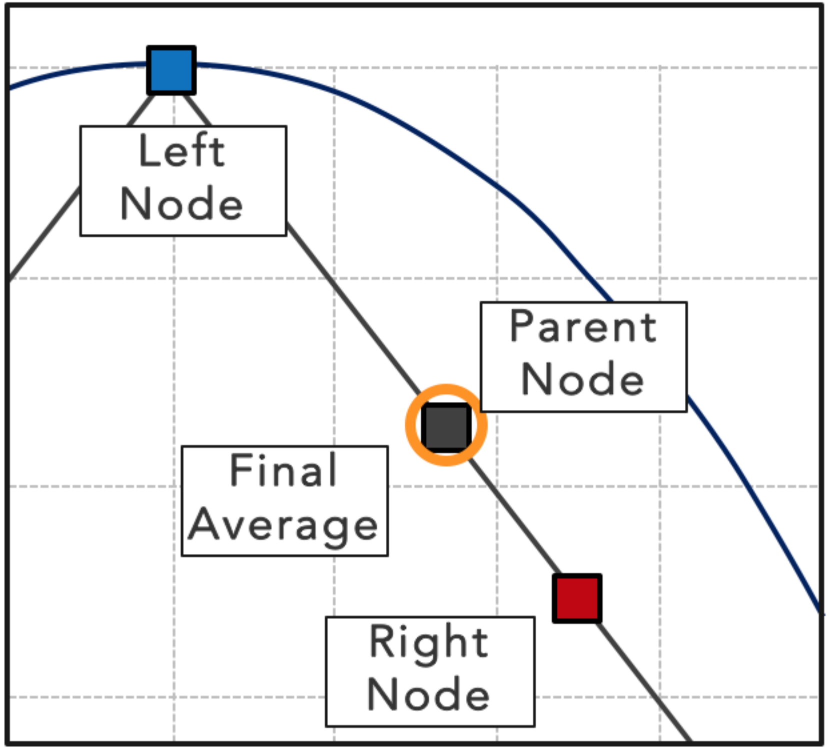
- With classification error, the function is flat

Information Gained by Splitting



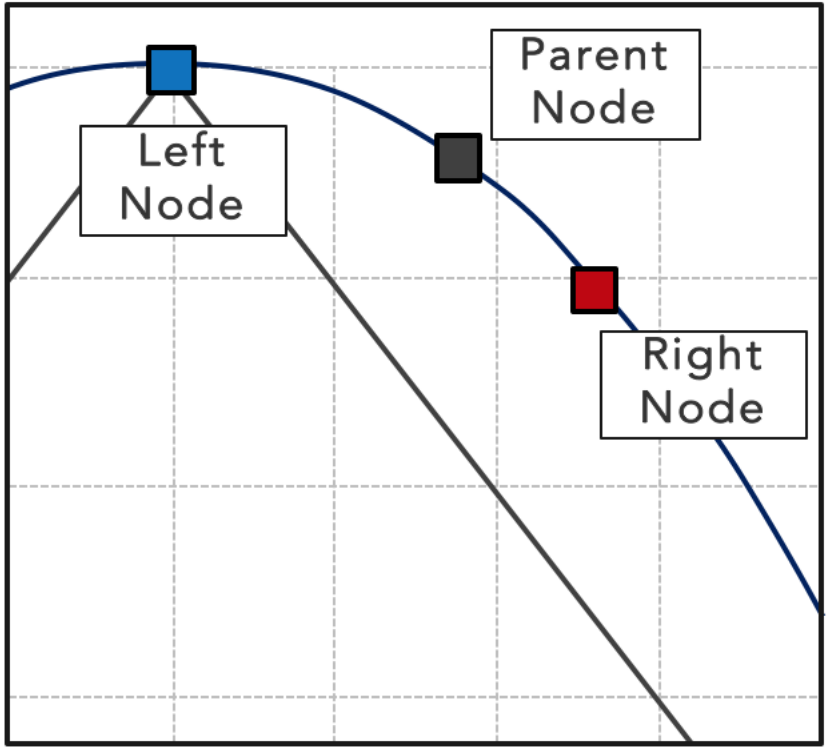
- With classification error, the function is flat
- Final average classification error can be identical to parent

Information Gained by Splitting



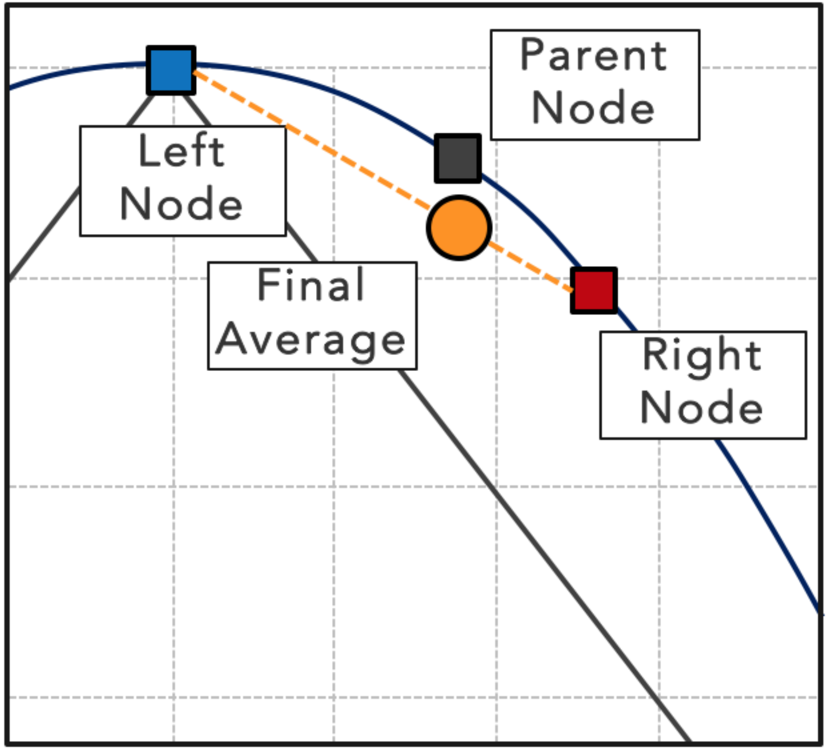
- With classification error, the function is flat
- Final average classification error can be identical to parent
- Resulting in premature stopping

Information Gained by Splitting



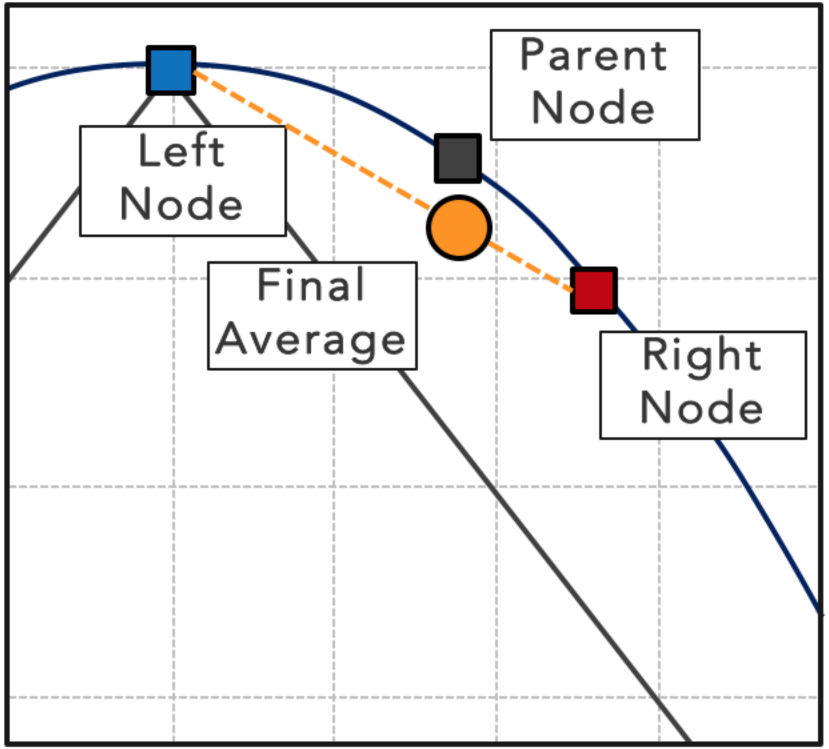
- With entropy gain, the function has a "bulge"

Information Gained by Splitting



- With entropy gain, the function has a "bulge"
- Allows average information of children to be less than parent
- Results in information gain and

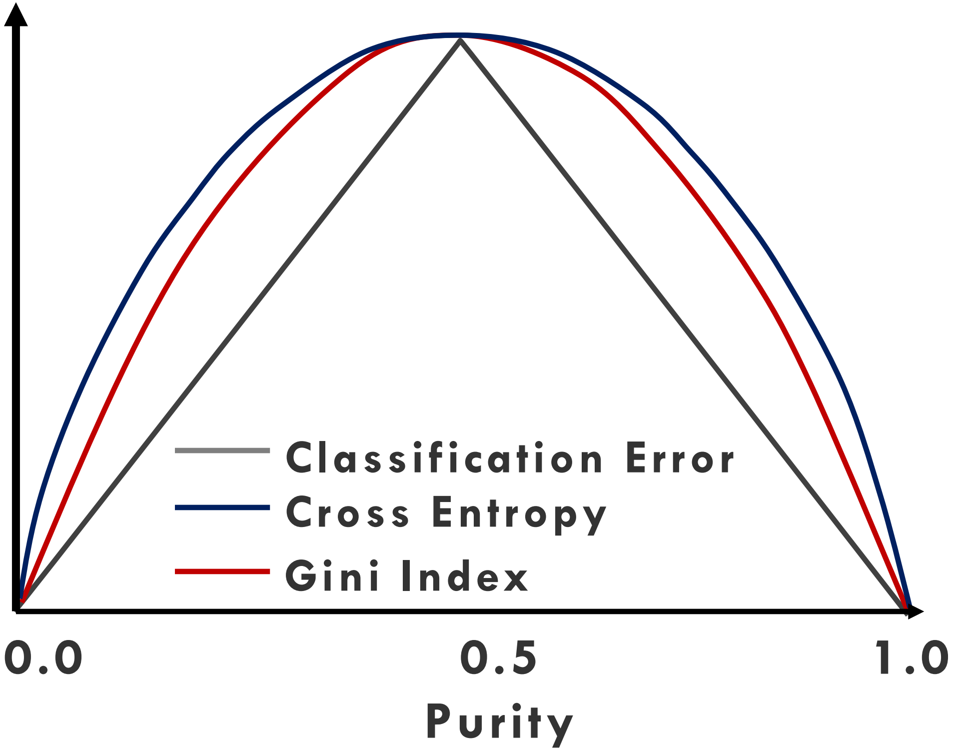
Information Gained by Splitting



- With entropy gain, the function has a "bulge"
- Allows average information of children to be less than parent
- Results in information gain and continued splitting

The Gini Index

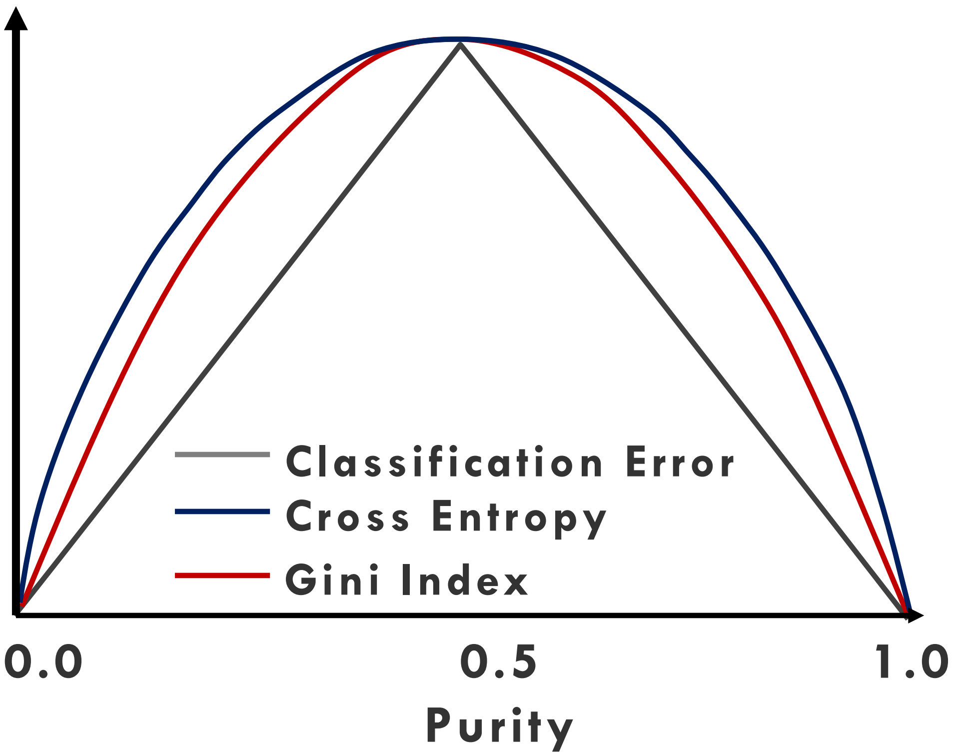
- In practice, Gini index often used for splitting



$$G(t) = 1 - \sum_{i=1}^{n} p(i|t)^2$$

The Gini Index

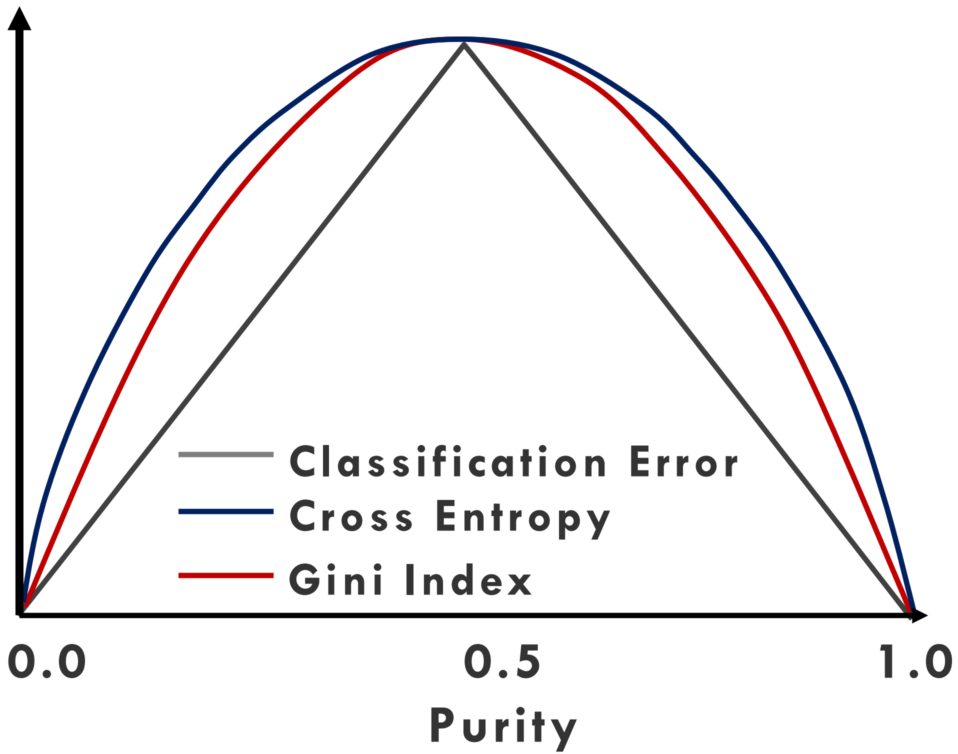
- In practice, Gini index often used for splitting
- Function is similar to entropy—has bulge



$$G(t) = 1 - \sum_{i=1}^{n} p(i|t)^2$$

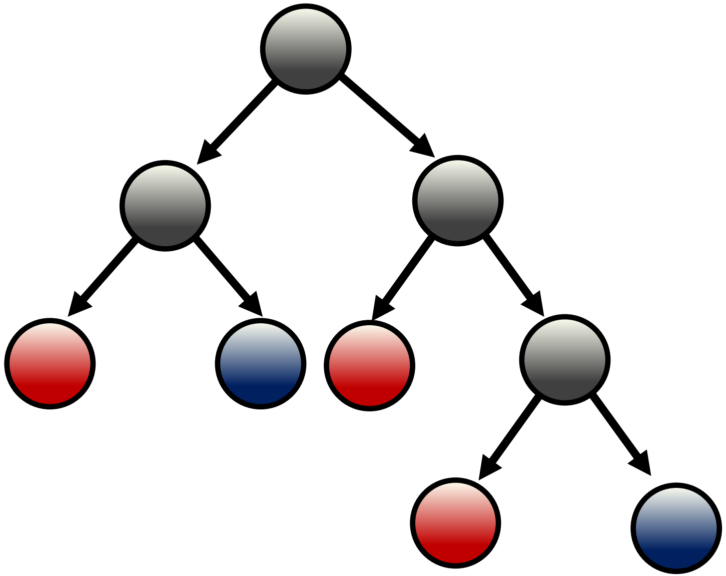
The Gini Index

- In practice, Gini index often used for splitting
- Function is similar to entropy—has bulge
- Does not contain logarithm



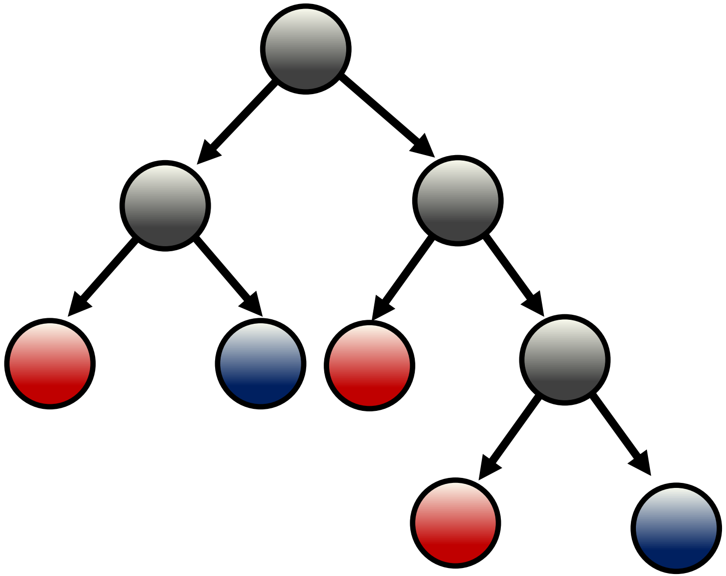
$$G(t) = 1 - \sum_{i=1}^{n} p(i|t)^2$$

Decision Trees are High Variance



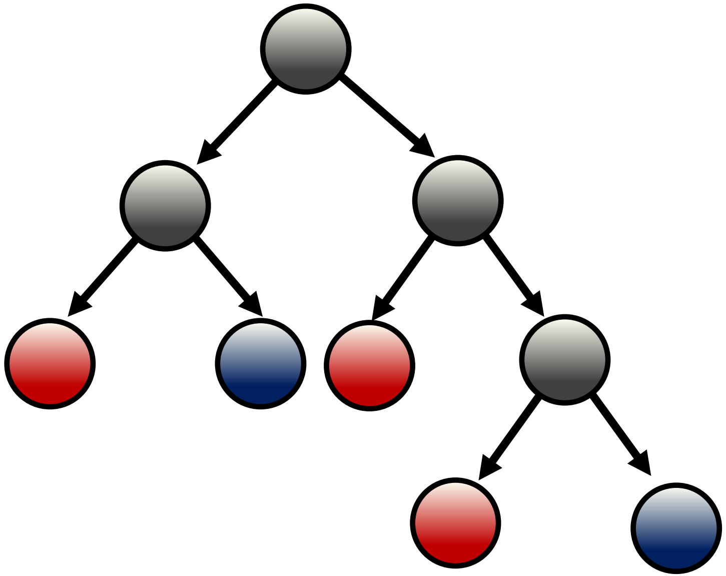
- Problem: decision trees tend to overfit

Decision Trees are High Variance



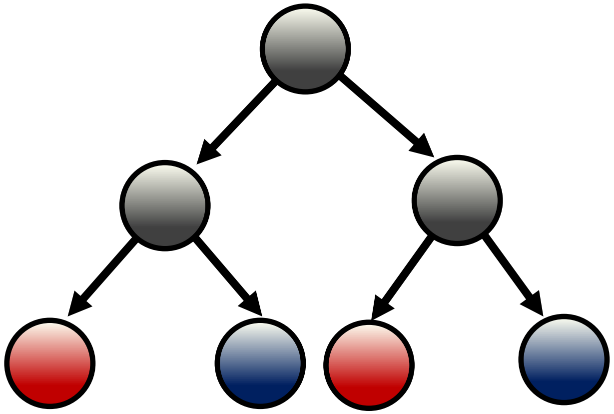
- Problem: decision trees tend to overfit
- Small changes in data greatly affect prediction--high variance
- Solution: Prune trees

Decision Trees are High Variance



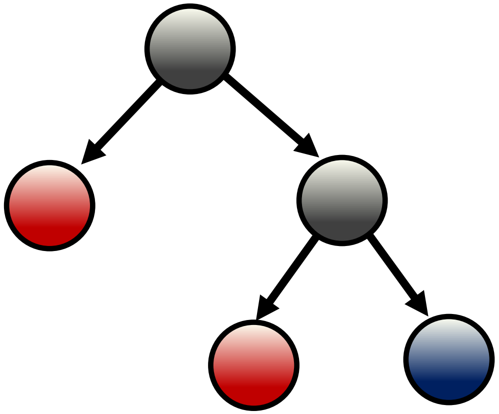
- Problem: decision trees tend to overfit
- Small changes in data greatly affect prediction--high variance
- Solution: Prune trees

Pruning Decision Trees



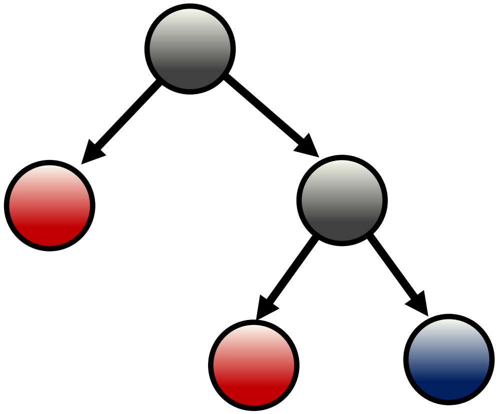
- Problem: decision trees tend to overfit
- Small changes in data greatly affect prediction--high variance
- Solution: Prune trees

Pruning Decision Trees



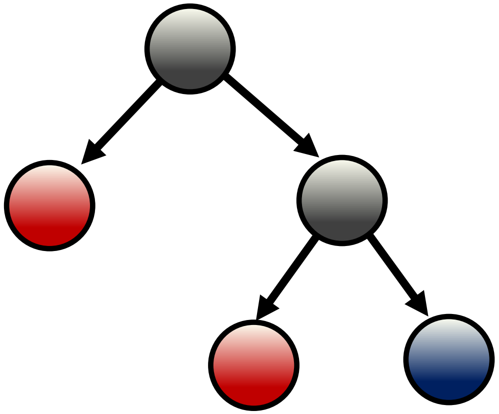
- Problem: decision trees tend to overfit
- Small changes in data greatly affect prediction--high variance
- Solution: Prune trees

Pruning Decision Trees



- How to decide what leaves to prune?

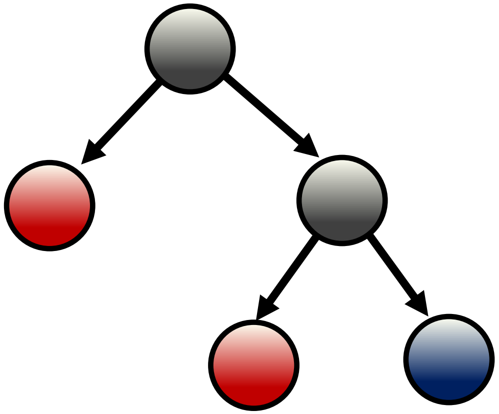
Pruning Decision Trees



- How to decide what leaves to prune?
- Solution: prune based on classification error threshold

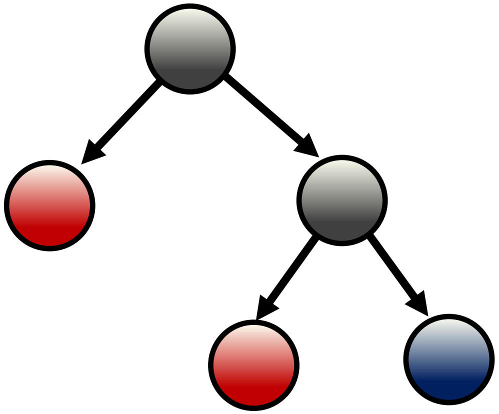
$$E(t) = 1 - \max_i [p(i|t)]$$

Strengths of Decision Trees



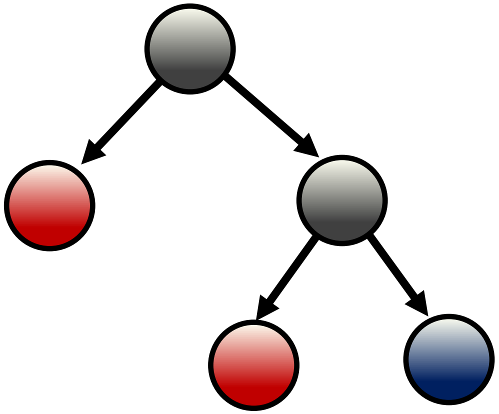
- Easy to interpret and implement—"if … then … else" logic

Strengths of Decision Trees



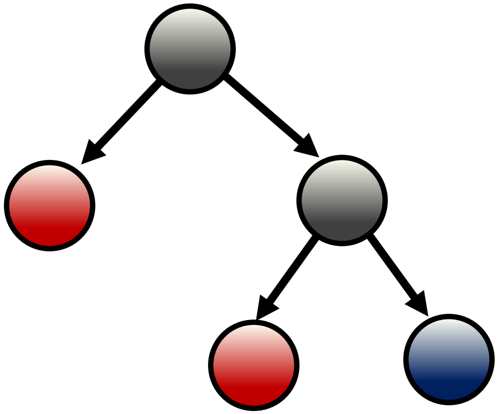
- Easy to interpret and implement—"if … then … else" logic
- Handle any data category—binary, ordinal, continuous

Strengths of Decision Trees



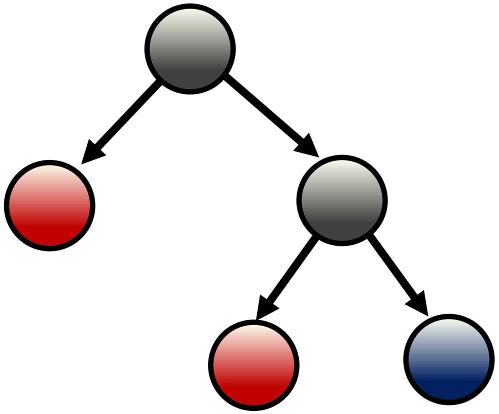
- Easy to interpret and implement—"if … then … else" logic
- Handle any data category—binary, ordinal, continuous
- No preprocessing or scaling required

Weakness of Decision Trees



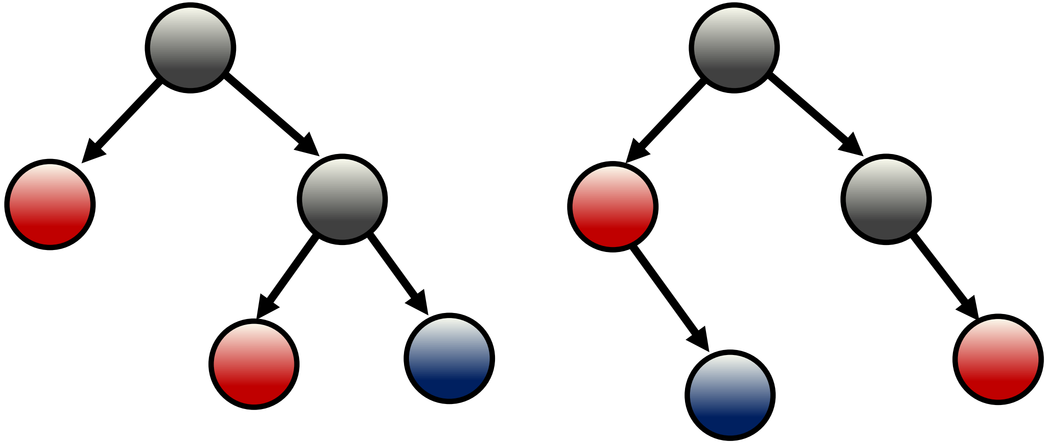
- Decision trees are less appropriate for continuous variable regression tasks.
- Decision trees are prone to overfitting if too deep.

Alternative to Decision Tree



- Almost any Scikit-learn classification algorithm
- Random Forest is the most conceptually similar, being a collection of decision trees.

Random Forest



- A collection of individual Decision Trees
- Aggregates many decision trees to limit overfitting
- Minimizes errors due to bias

DecisionTreeClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.tree import DecisionTreeClassifier
```

DecisionTreeClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
DTC = DecisionTreeClassifier(criterion='gini',
                             max_features=10, max_depth=5)
```

DecisionTreeClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
DTC = DecisionTreeClassifier(criterion='gini',
                             max_features=10, max_depth=5)
```

tree parameters

DecisionTreeClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
DTC = DecisionTreeClassifier(criterion='gini',
                             max_features=10, max_depth=5)
```

Fit the instance on the data and then predict the expected value

```
DTC = DTC.fit(X_train, y_train)
y_predict = DTC.predict(X_test)
```

DecisionTreeClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
DTC = DecisionTreeClassifier(criterion='gini',
                             max_features=10, max_depth=5)
```

Fit the instance on the data and then predict the expected value

```
DTC = DTC.fit(X_train, y_train)
y_predict = DTC.predict(X_test)
```

Tune parameters with cross-validation. Use DecisionTreeRegressor for regression.

RandomForestClassifier: The Syntax

To use the Intel® Extension for Scikit-learn* variant of this algorithm:

- **Install Intel® oneAPI AI Analytics Toolkit (AI Kit)**
- **Add the following two lines of code after the above code:**

```
from sklearnex import patch_sklearn
patch_sklearn().
```

Import the class containing the classification method

```
from sklearn.ensemble import RandomForestClassifier
```

intel®

Software