Contents

1. Understand Decision Trees
2. Discover entropy and information gain
3. Discuss decision tree limitations + applications

Decision Trees

- A decision tree is a tree-like model that is used for making decisions.
- It consists of nodes that represent decision points, and branches that represent the outcomes of those decisions.
- The decision points are based on the values of the input variables, and the outcomes are the possible classifications or regression predictions.

https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

Decision Trees for Classification

Example: The car dataset

cylinders	displacement	horsepower	weight	acceleration	year	maker	mpg
4	112	88	2395	18	82	America	Good
4	135	84	2370	13	82	America	Bad
4	135	84	2370	13	82	America	Good
4	91	68	1970	17.6	82	Europe	Good
4	105	63	2125	14.7	82	America	Bad
4	105	63	2125	14.7	82	America	Good
6	146	120	2930	13.8	81	Europe	OK
4	156	92	2620	14.4	81	America	OK
4	81	60	1760	16.1	81	Europe	Good
4	140	88	2870	18.1	80	America	OK
8	305	130	3840	15.4	79	America	Bad
6	200	85	2990	18.2	79	America	Bad

Full dataset link here!

Example: The car dataset

Split data into a tree structure

- Let's split the dataset on how heavy the car is.
- The root note represents entire dataset (19 bad, 15 okay, and 8 good examples)
- The leaves have some subset of bad/okay/good
  - Some leaves are **pure** (all bad, okay, or good)
  - Some leaves are **mixed** → split further

Example: The car dataset

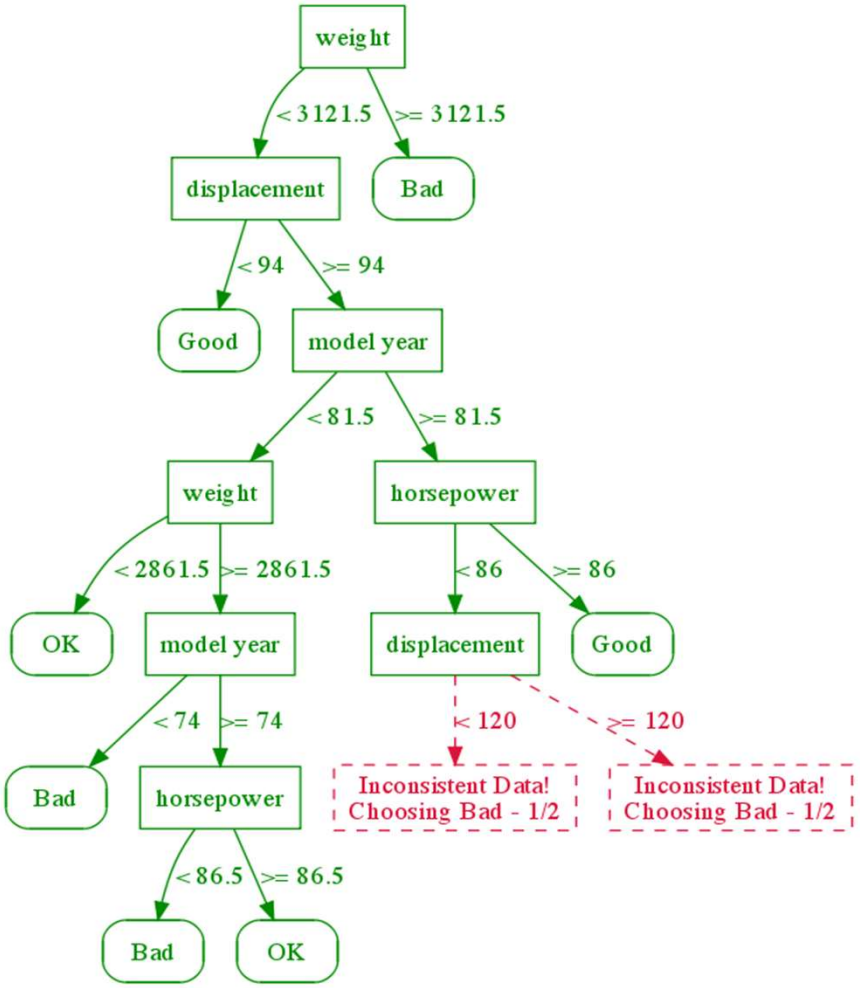
Second Split

- Let's split the dataset on displacement
- A leaf is now pure (good!)
- Another leaf is still mixed → split further

Example: The car dataset

Keep Splitting

- Keep on splitting until all leaves are pure
- Sometimes this actually might not be possible (see the red leaves 1 out of 2 examples is bad)
- We can use this organization of the data as a classifier
- Given a test example, we traverse the tree from root to the leaf and return the leaf label → **decision tree!**



Decision Trees

- Like SVMs, Decision Tree (DT) is a versatile ML algorithm for classification and regression (Recall **DecisionTreeRegressor**)
- We will discuss how to train, visualize, and predict with Decision Trees, including Classification and Regression Tree (**CART)**.
- Later, it is a fundamental component of **Random Forest**

The CART Training Algorithm

1. Calculate the impurity for the entire dataset. This is the impurity of the root node.
2. For each input variable, calculate the impurity for all possible split points. The split point that results in the minimum impurity is chosen.
3. The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
4. Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
5. The resulting tree is the decision tree.

https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

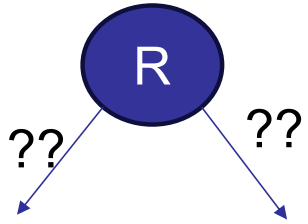
The CART Training Algorithm

1. Calculate the impurity for the entire dataset. This is the impurity of the root node.
2. For each input variable, calculate the impurity for all possible split points. The split point that results in the minimum impurity is chosen.
3. The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
4. Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
5. The resulting tree is the decision tree.

https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

The CART Training Algorithm

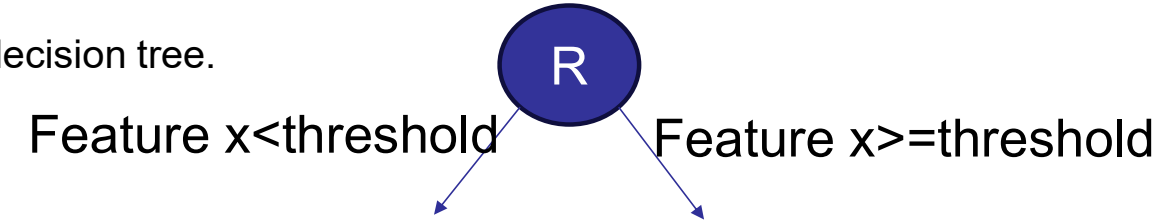
1. Calculate the impurity for the entire dataset. This is the impurity of the root node.
2. For each input variable, calculate the impurity for all possible split points. The split point that results in the minimum impurity is chosen.
3. The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
4. Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
5. The resulting tree is the decision tree.



https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

The CART Training Algorithm

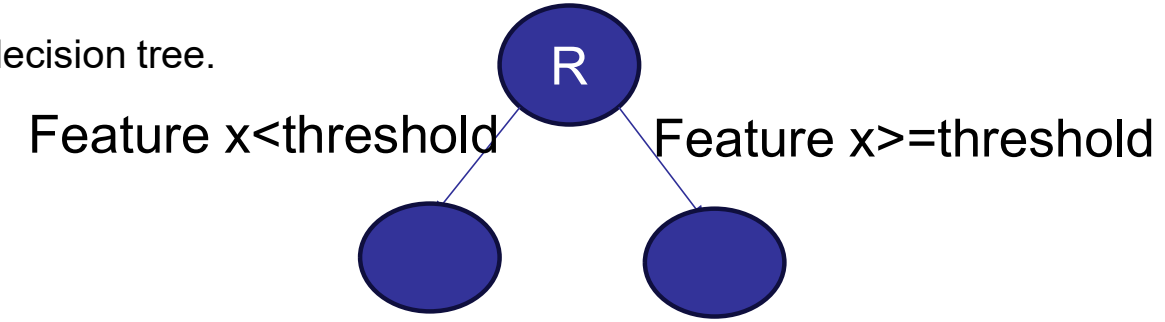
1. Calculate the impurity for the entire dataset. This is the impurity of the root node.
2. For each input variable, calculate the impurity for all possible split points. The split point that results in the minimum impurity is chosen.
3. The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
4. Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
5. The resulting tree is the decision tree.



https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

The CART Training Algorithm

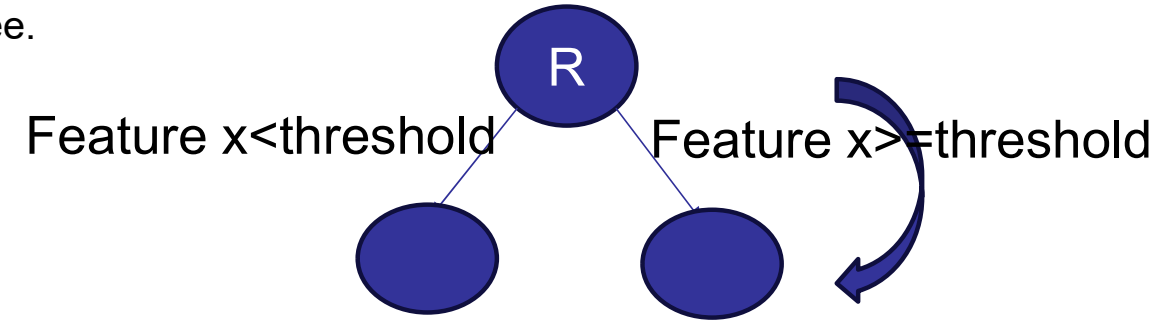
1. Calculate the impurity for the entire dataset. This is the impurity of the root node.
2. For each input variable, calculate the impurity for all possible split points. The split point that results in the minimum impurity is chosen.
3. The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
4. Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
5. The resulting tree is the decision tree.



https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

The CART Training Algorithm

1. Calculate the impurity for the entire dataset. This is the impurity of the root node.
2. For each input variable, calculate the impurity for all possible split points. The split point that results in the minimum impurity is chosen.
3. The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
4. Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
5. The resulting tree is the decision tree.



https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

CART

- Stopping Criteria
  - Maximum depth
  - Minimum number of instances in each leaf node
  - Feature values for all records are identical
  - Other criteria

How to measure impurity?!

- We need to define a way for measuring **impurity** to select features and splitting thresholds

Gini index measures the probability of a random instance being misclassified when chosen randomly

Entropy measures disorder, randomness, or uncertainty

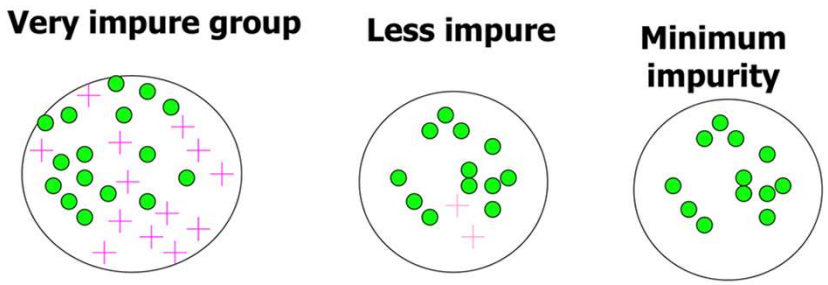
Decision Trees: Features

Choosing what feature to split on

A node is "pure" if all samples it applies to belong to the same class (or its impunity = 0). Otherwise its impurity (**Gini index**) is computed as:

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

$p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node.

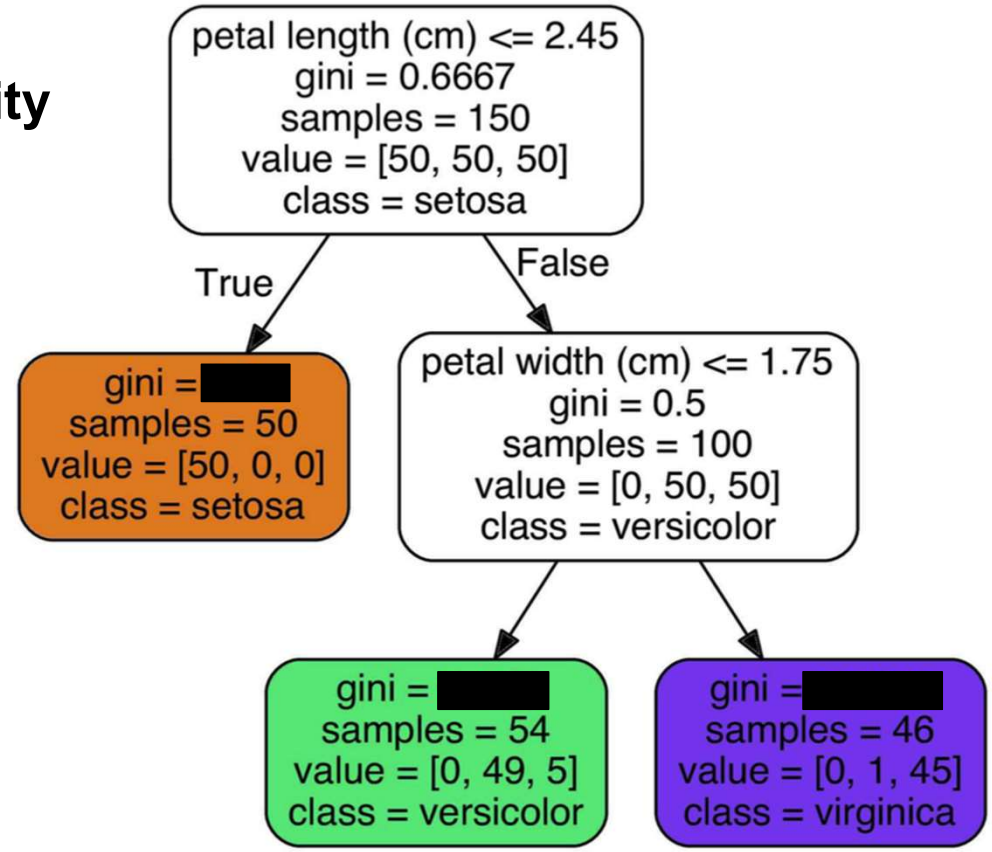


Gini index measures the probability of a random instance being misclassified when chosen randomly

Decision Trees: Features

Compute Gini Impurity

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$



Decision Trees: Features

Estimating Class Probability

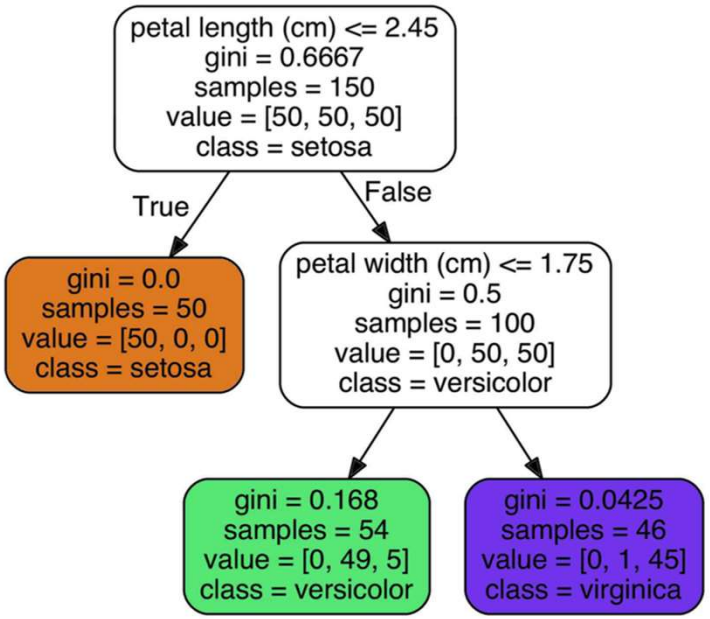
- Predict by estimating probability that a sample belongs to a particular class k
- Traverse the tree to leaf node, then return the ratio of training samples of class k

```
tree_clf.predict_proba([[5, 1.5]])
```

```
array([[ 0.        ,  0.90740741,  0.09259259]])
```

```
tree_clf.predict([[5, 1.5]])
```

```
array([1])
```



D

E

- Pr
sa
- Tr
re
cl

predict_proba(*X, check_input=True*) [source]

Predict class probabilities of the input samples X.

The predicted class probability is the fraction of samples of the same class in a leaf.

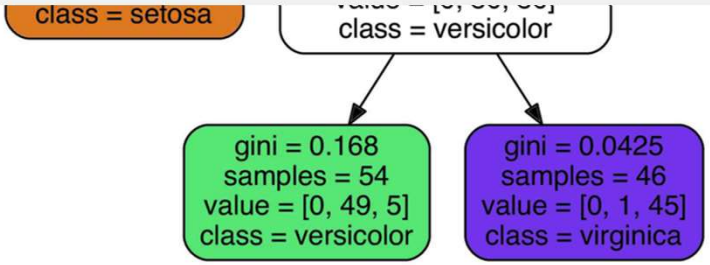
Parameters:	**X :** ***{array-like, sparse matrix} of shape (n_samples, n_features)*** The input samples. Internally, it will be converted to `dtype=np.float32` and if a sparse matrix is provided to a sparse `csr_matrix`. **check_input :** ***bool, default=True*** Allow to bypass several input checking. Don't use this parameter unless you know what you're doing.
Returns:	**proba :** ***ndarray of shape (n_samples, n_classes) or list of n_outputs such arrays if n_outputs > 1*** The class probabilities of the input samples. The order of the classes corresponds to that in the attribute classes_.

```
tree_clf.predict_proba([[5, 1.5]])
```

```
array([[ 0.        ,  0.90740741,  0.09259259]])
```

```
tree_clf.predict([[5, 1.5]])
```

```
array([1])
```



Deci

Estin

- Predic
  sampl
- Trave
  return
  class k

predict(*X, check_input=True*) [source]

Predict class or regression value for X.

For a classification model, the predicted class for each sample in X is returned. For a regression model, the predicted value based on X is returned.

Parameters:

X : *{array-like, sparse matrix} of shape (n_samples, n_features)*
The input samples. Internally, it will be converted to `dtype=np.float32` and if a sparse matrix is provided to a sparse `csr_matrix`.

check_input : *bool, default=True*
Allow to bypass several input checking. Don't use this parameter unless you know what you're doing.

Returns:

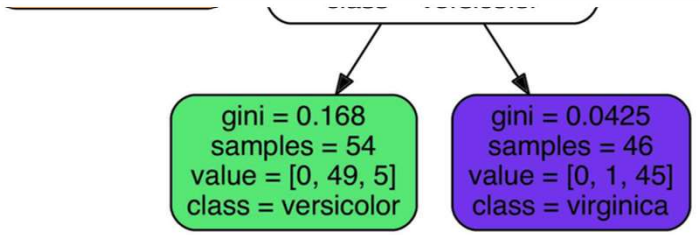
y : *array-like of shape (n_samples,) or (n_samples, n_outputs)*
The predicted classes, or the predict values.

```
tree_clf.predict_proba([[5, 1.5]])
```

```
array([[ 0.        ,  0.90740741,  0.09259259]])
```

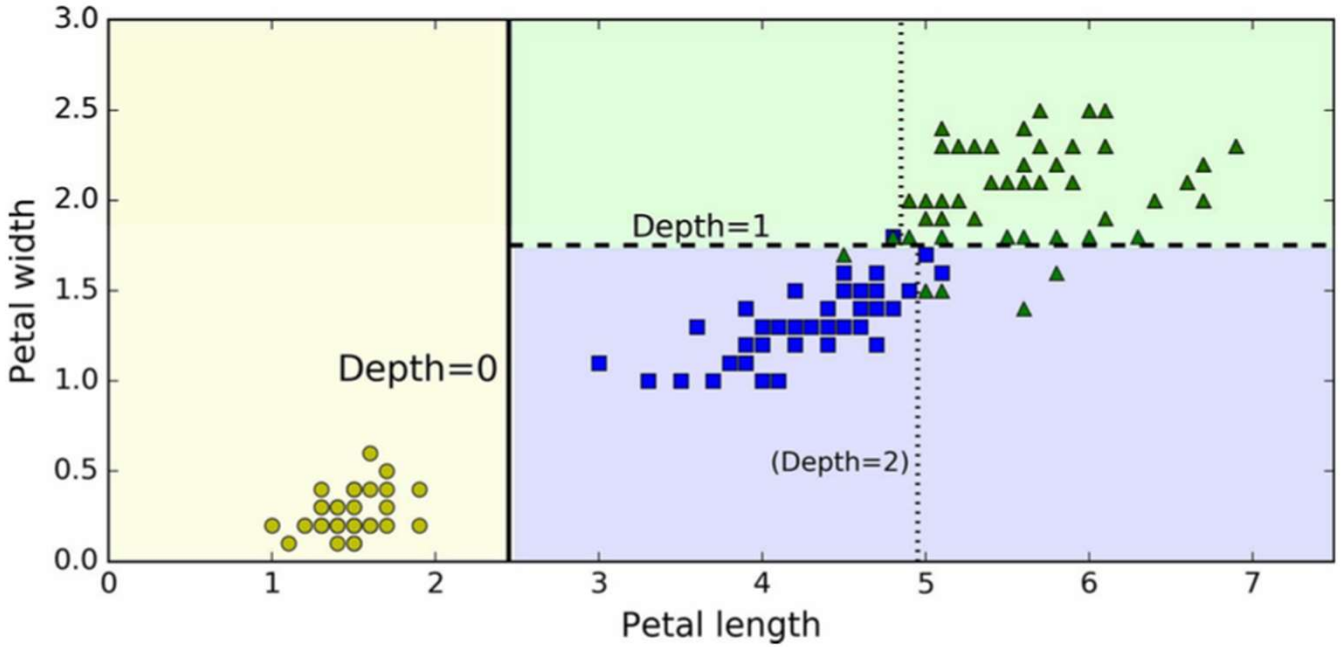
```
tree_clf.predict([[5, 1.5]])
```

```
array([1])
```



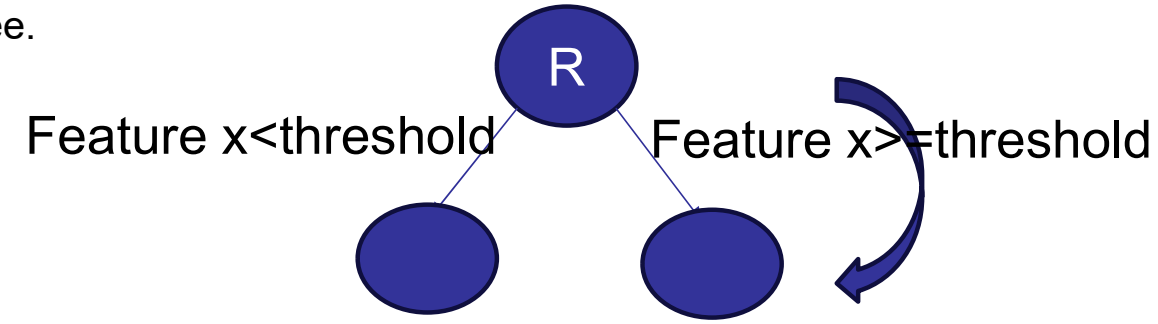
Decision Trees: Features

Decision Boundary



The CART Training Algorithm

1. Calculate the impurity for the entire dataset. This is the impurity of the root node.
2. For each input variable, calculate the impurity for all possible split points. The split point that results in the minimum impurity is chosen.
3. The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
4. Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
5. The resulting tree is the decision tree.



https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

Decision Trees: CART

The CART Training Algorithm

The Classification and Regression Tree (CART) algorithm to train Decision Trees

- Search for the pair of a single feature and its threshold (k, t_k) that produce the subset with **lowest Gini index:**

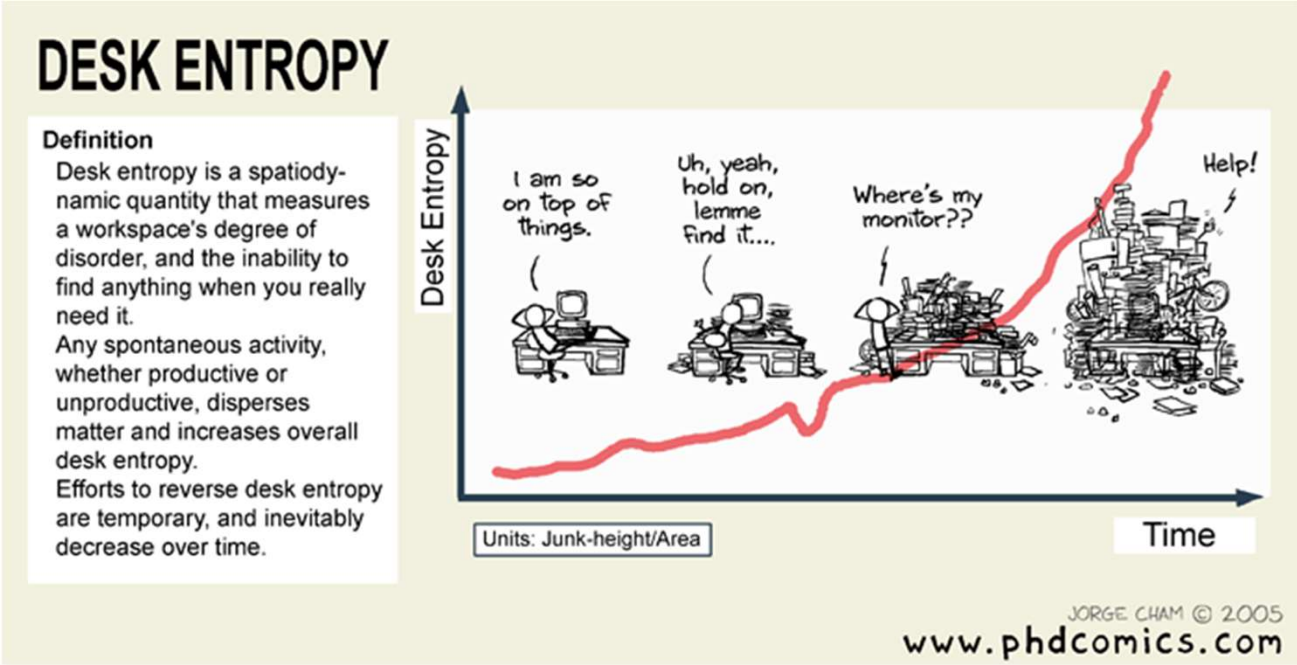
$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

where $\begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$

- Split training set into two subsets using feature k and threshold t_k
- Split the subset using the same logic, stop once reaches max depth

Decision Trees: Features

Entropy, anyone?



Entropy measures disorder, randomness, or uncertainty

Decision Trees: Features

Entropy

- Entropy is a measure of **disorderness**: it is zero when well-order and identical
- A set's entropy is zero when it contains instances of only one class (same as Gini)
- Entropy is calculated as H_i:

$$H_i = -\sum_{k=1}^{n} p_{i,k} \log(p_{i,k})$$

Decision Trees: Features

Entropy Interpretation

For binary Y, the entropy is a function of **p(y)**:

$$H(Y) = -\sum_{y \in Y} p(y) \log p(y)$$

Note: entropy is 0 when **p=1 or p=0** (all are of the same class)

Entropy is 1 when **p=0.5** (50% in either class)

.

Decision Trees: Features

Information Gain

- To measure the reduction in entropy of Y from knowing X, we use Information Gain:

$$\mathrm{IG}\,(Y, X) = H(Y) - H(Y|X)$$

$$\text{where } H(Y|X) = \sum_{x \in X} p(x) H(Y|X = x)$$

- To determine which feature in a given set of feature vectors is most useful to discriminating between the classes
- We will use IG to decide the ordering of features in the nodes of a decision tree.

Example: When to play tennis?

Example: when to play tennis?

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	High	True	***No***

The CART Training Algorithm

1. Calculate the impurity for the entire dataset. This is the impurity of the root node.
2. For each input variable, calculate the impurity for all possible split points. The split point that results in the minimum impurity is chosen.
3. The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
4. Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
5. The resulting tree is the decision tree.

https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

Example: When to play tennis?

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	*No*
Sunny	Hot	High	True	*No*
Overcast	Hot	High	False	*Yes*
Rainy	Mild	High	False	*Yes*
Rainy	Cool	Normal	False	*Yes*
Rainy	Cool	Normal	True	*No*
Overcast	Cool	Normal	True	*Yes*
Sunny	Mild	High	False	*No*
Sunny	Cool	Normal	False	*Yes*
Rainy	Mild	Normal	False	*Yes*
Sunny	Mild	Normal	True	*Yes*
Overcast	Mild	High	True	*Yes*
Overcast	Hot	Normal	False	*Yes*
Rainy	Mild	High	True	*No*

$$\mathrm{H}(Y) = -\sum_{i=1}^{K} p_k \log_2 p_k$$

$$= -\frac{5}{14}\log_2\frac{5}{14} - \frac{9}{14}\log_2\frac{9}{14}$$

$$= 0.94$$

The CART Training Algorithm

1. Calculate the impurity for the entire dataset. This is the impurity of the root node.
2. For each input variable, calculate the impurity for all possible split points. The split point that results in the minimum impurity is chosen.
3. The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
4. Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
5. The resulting tree is the decision tree.

https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

Example: When to play tennis?

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	*No*
Sunny	Hot	High	True	*No*
Overcast	Hot	High	False	*Yes*
Rainy	Mild	High	False	*Yes*
Rainy	Cool	Normal	False	*Yes*
Rainy	Cool	Normal	True	*No*
Overcast	Cool	Normal	True	*Yes*
Sunny	Mild	High	False	*No*
Sunny	Cool	Normal	False	*Yes*
Rainy	Mild	Normal	False	*Yes*
Sunny	Mild	Normal	True	*Yes*
Overcast	Mild	High	True	*Yes*
Overcast	Hot	Normal	False	*Yes*
Rainy	Mild	High	True	*No*

$$InfoGain(\text{Humidity}) = H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$
$$= 0.94 - \frac{7}{14} H_L - \frac{7}{14} H_R$$

Example: When to play tennis?

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	**Normal**	False	***Yes***
Rainy	Cool	**Normal**	True	***No***
Overcast	Cool	**Normal**	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	**Normal**	False	***Yes***
Rainy	Mild	**Normal**	False	***Yes***
Sunny	Mild	**Normal**	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	**Normal**	False	***Yes***
Rainy	Mild	High	True	***No***

$$InfoGain(\text{Humidity}) = H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

$$0.94 - \frac{7}{14} H_L - \frac{7}{14} H_R$$

$$H_L = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7}$$

Example: When to play tennis?

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	**High**	False	*No*
Sunny	Hot	**High**	True	*No*
Overcast	Hot	**High**	False	*Yes*
Rainy	Mild	**High**	False	*Yes*
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	**High**	False	*No*
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	**High**	True	*Yes*
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	**High**	True	*No*

$$InfoGain(\text{Humidity}) = H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$

$$0.94 - \frac{7}{14} H_L - \frac{7}{14} H_R$$

$$H_L = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.592$$

$$H_R = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.985$$

Example: When to play tennis?

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	***No***
Sunny	Hot	High	True	***No***
Overcast	Hot	High	False	***Yes***
Rainy	Mild	High	False	***Yes***
Rainy	Cool	Normal	False	***Yes***
Rainy	Cool	Normal	True	***No***
Overcast	Cool	Normal	True	***Yes***
Sunny	Mild	High	False	***No***
Sunny	Cool	Normal	False	***Yes***
Rainy	Mild	Normal	False	***Yes***
Sunny	Mild	Normal	True	***Yes***
Overcast	Mild	High	True	***Yes***
Overcast	Hot	Normal	False	***Yes***
Rainy	Mild	High	True	***No***

$$InfoGain(\text{Humidity}) = H(Y) - \frac{m_L}{m} H_L - \frac{m_R}{m} H_R$$
$$0.94 - \frac{7}{14} 0.592 - \frac{7}{14} 0.985$$
$$= 0.94 - 0.296 - 0.4925$$
$$= 0.1515$$

Example: When to play tennis?

Calculating for each feature:

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	*No*
Sunny	Hot	High	True	*No*
Overcast	Hot	High	False	*Yes*
Rainy	Mild	High	False	*Yes*
Rainy	Cool	Normal	False	*Yes*
Rainy	Cool	Normal	True	*No*
Overcast	Cool	Normal	True	*Yes*
Sunny	Mild	High	False	*No*
Sunny	Cool	Normal	False	*Yes*
Rainy	Mild	Normal	False	*Yes*
Sunny	Mild	Normal	True	*Yes*
Overcast	Mild	High	True	*Yes*
Overcast	Hot	Normal	False	*Yes*
Rainy	Mild	High	True	*No*

`IG(Humidity) = 0.1515`

`IG(Outlook) = 0.247`

`IG(Temperature) = 0.029`

`IG(Windy) = 0.048`

→ Initial split is on **Outlook** because it is the feature with the highest `IG`.

The CART Training Algorithm

1. Calculate the impurity for the entire dataset. This is the impurity of the root node.
2. For each input variable, calculate the impurity for all possible split points. The split point that results in the minimum impurity is chosen.
3. The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
4. Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
5. The resulting tree is the decision tree.

https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

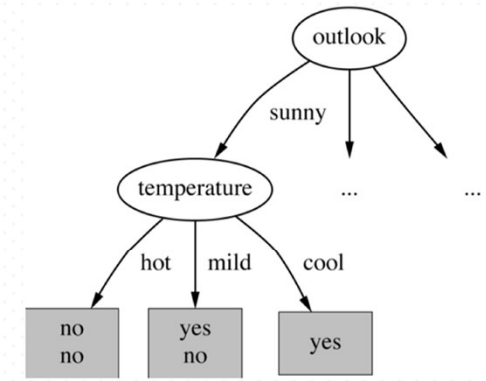
The CART Training Algorithm

1. Calculate the impurity for the entire dataset. This is the impurity of the root node.
2. For each input variable, calculate the impurity for all possible split points. The split point that results in the minimum impurity is chosen.
3. The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
4. Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
5. The resulting tree is the decision tree.

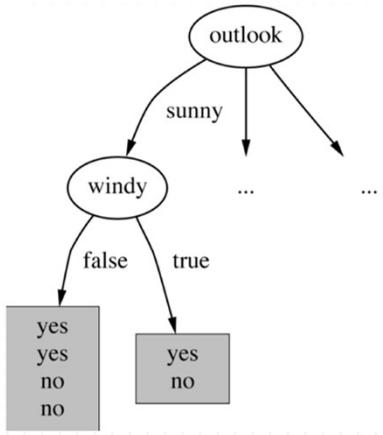
https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

Example: When to play tennis?

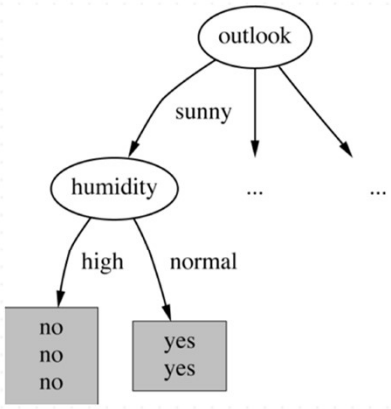
Search for best split at the next level:



Temperature = 0.571



Windy = 0.020



Humidity = 0.971

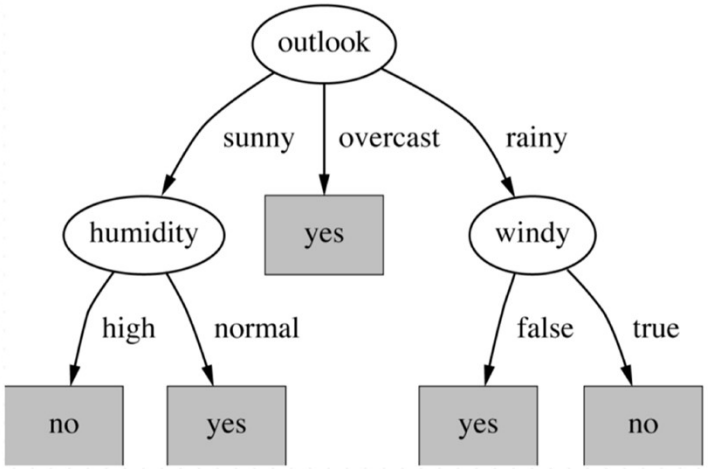
The CART Training Algorithm

1. Calculate the impurity for the entire dataset. This is the impurity of the root node.
2. For each input variable, calculate the impurity for all possible split points. The split point that results in the minimum impurity is chosen.
3. The data is split into two subsets based on the chosen split point, and a new node is created for each subset.
4. Steps 2 and 3 are repeated for each new node, until a stopping criterion is met. This stopping criterion could be a maximum tree depth, a minimum number of data points in a leaf node, or a minimum reduction in impurity.
5. The resulting tree is the decision tree.

https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf

Example: When to play tennis?

The final decision tree



Note that not all leaves need to be pure; sometimes similar (even identical) instances have different classes. Splitting stops when data cannot be split any further.

How to measure impurity?!

- We need to define a way for measuring **impurity** to select features and splitting thresholds

Gini index measures the probability of a random instance being misclassified when chosen randomly

Entropy measures disorder, randomness, or uncertainty

- Both Entropy and Gini produce similar tree most of the time, but Gini is slightly faster.
- Gini is for numerical features while Entropy is for categorical ones.
- Gini tends to isolate the **largest** class from other classes while Entropy tends to find group that made up ~50% of the data → **more balanced tree**.

Computational Complexity

- Finding the optimal tree is known as a NP-Complete problem, so we must settle for a "reasonably good" solution.
- Traversing requires roughly **O**(log(m)) independent of number of features → very fast (m is the number of samples)
- Training comparing all features on all samples at each node **O**(nm log(m)) → slow down considerably for large training set. (n is the number of features)

What makes a good tree?

- **Nt too small:** need to handle important but subtle distinction in data
- **Not too big:** avoid overfitting training examples
- **Occam's Razor:** find the simplest model (smallest tree) that fits the observations
- **Regularization**: In practice, you can **regularize** the construction process to get small but effective tree

Occam's Razor: if you have two competing ideas to explain the same phenomenon, you should prefer the simpler one.

Regularization Hyperparameters

Left unconstrained, decision tree will most likely overfitting to the training data.

Need to restrict the tree degree of freedom → **regularization**

- **max_depth**: the maximum depth of the tree
- **min_sample_split**: the minimum number of samples a node must have to split
- **min_sample_leaf**: the minimum number of samples a leaf node must have
- **max_leaf_nodes:** maximum number of leaf nodes
- **max_features:** maximum number features that are evaluated for splitting

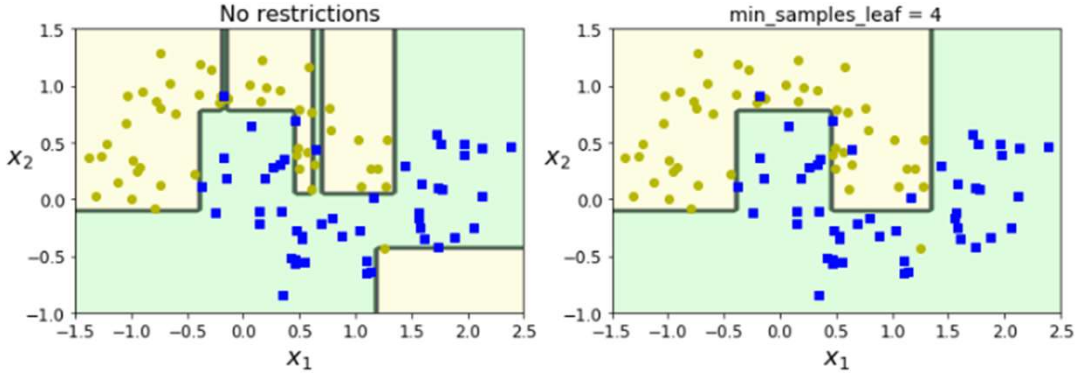
Training a Decision Tree in Python

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier

iris = load_iris()
X = iris.data[:, 2:] # petal length and width
y = iris.target

tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf.fit(X, y)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=2,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=42,
            splitter='best')
```

Code Demo

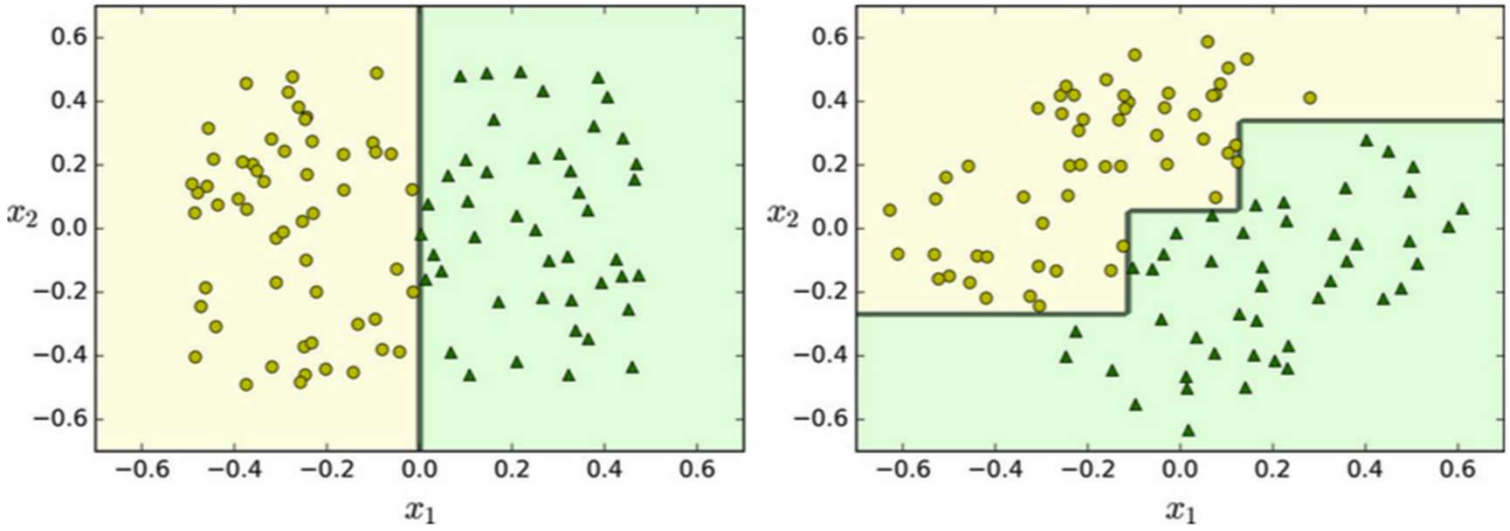
```
from sklearn.datasets import make_moons
Xm, ym = make_moons(n_samples=100, noise=0.25, random_state=53)

deep_tree_clf1 = DecisionTreeClassifier(random_state=42)
deep_tree_clf2 = DecisionTreeClassifier(min_samples_leaf=4, random_state=42)
deep_tree_clf1.fit(Xm, ym)
deep_tree_clf2.fit(Xm, ym)
```



Limitations and Applications

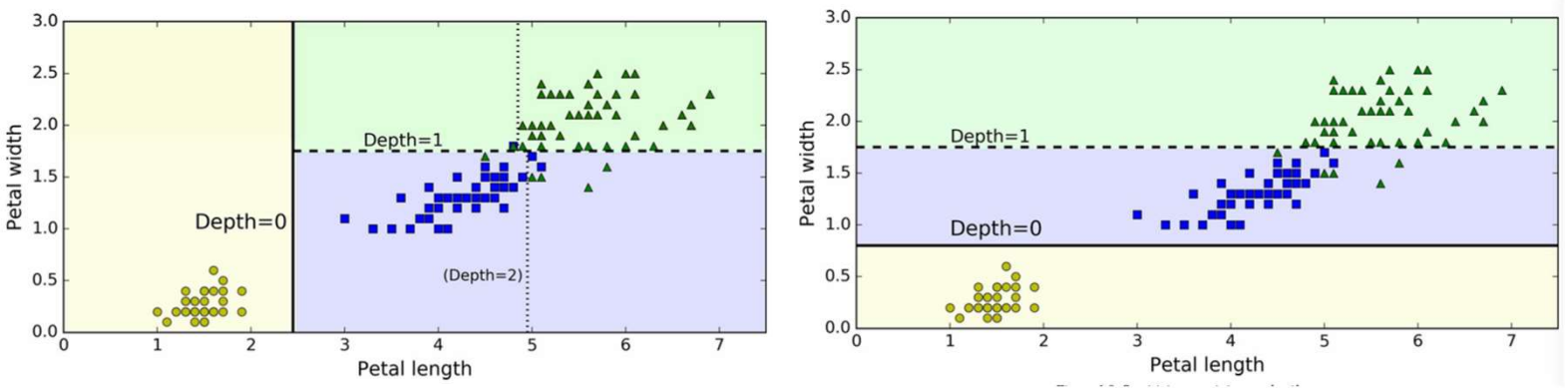
Sensitivity to Training Set

DT loves **orthogonal** decision boundary → sensitive to training set rotation



Limitations and Applications

Instability to small variation of training data

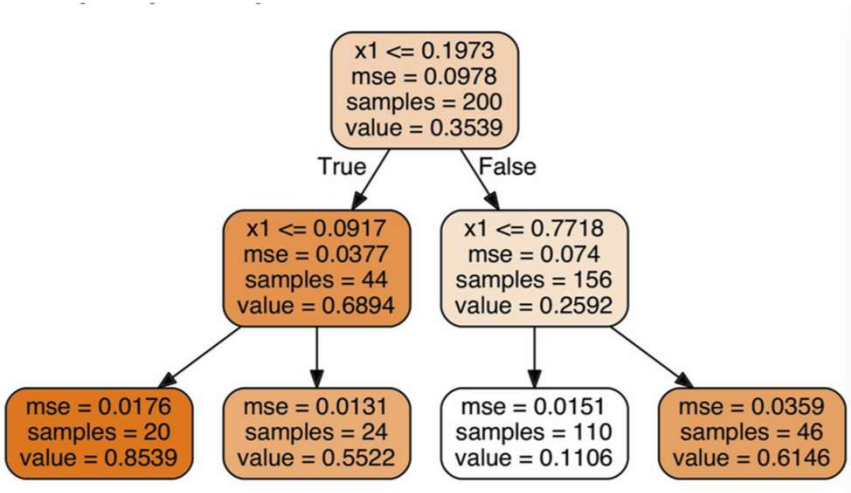


Random Forest might help!

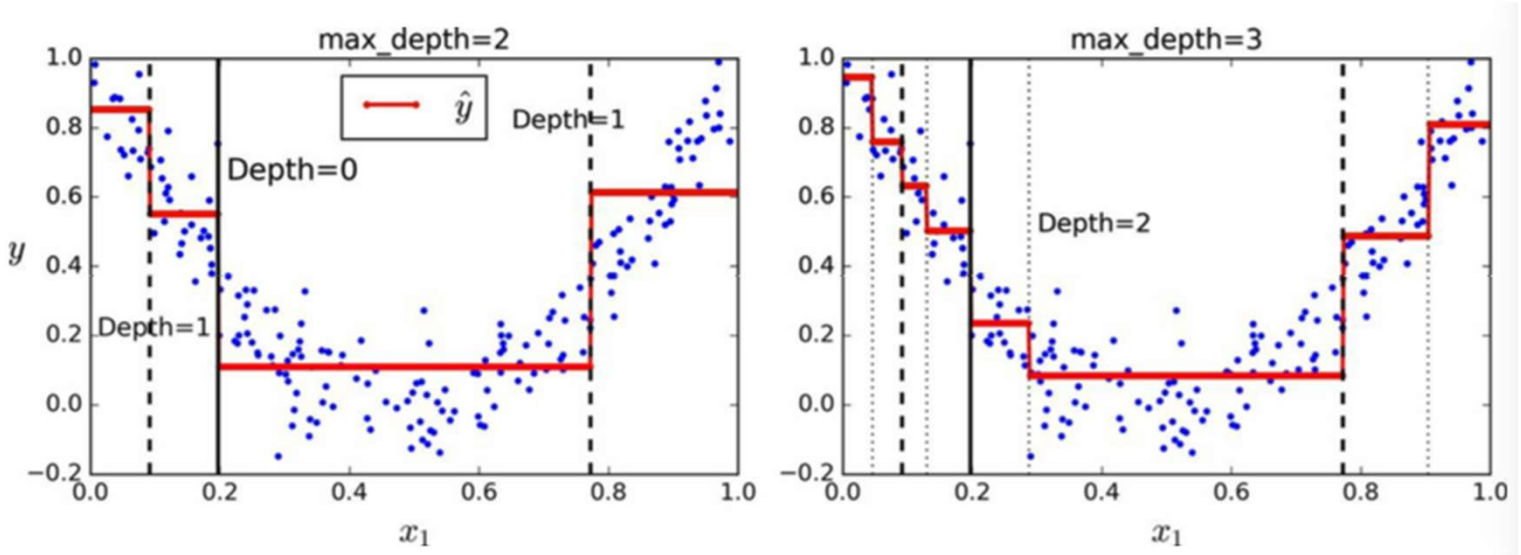
Decision Trees: Extensions

1. Regression

```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor(max_depth=2, random_state=42)
tree_reg.fit(X, y)
```



2. Max Depth and Prediction

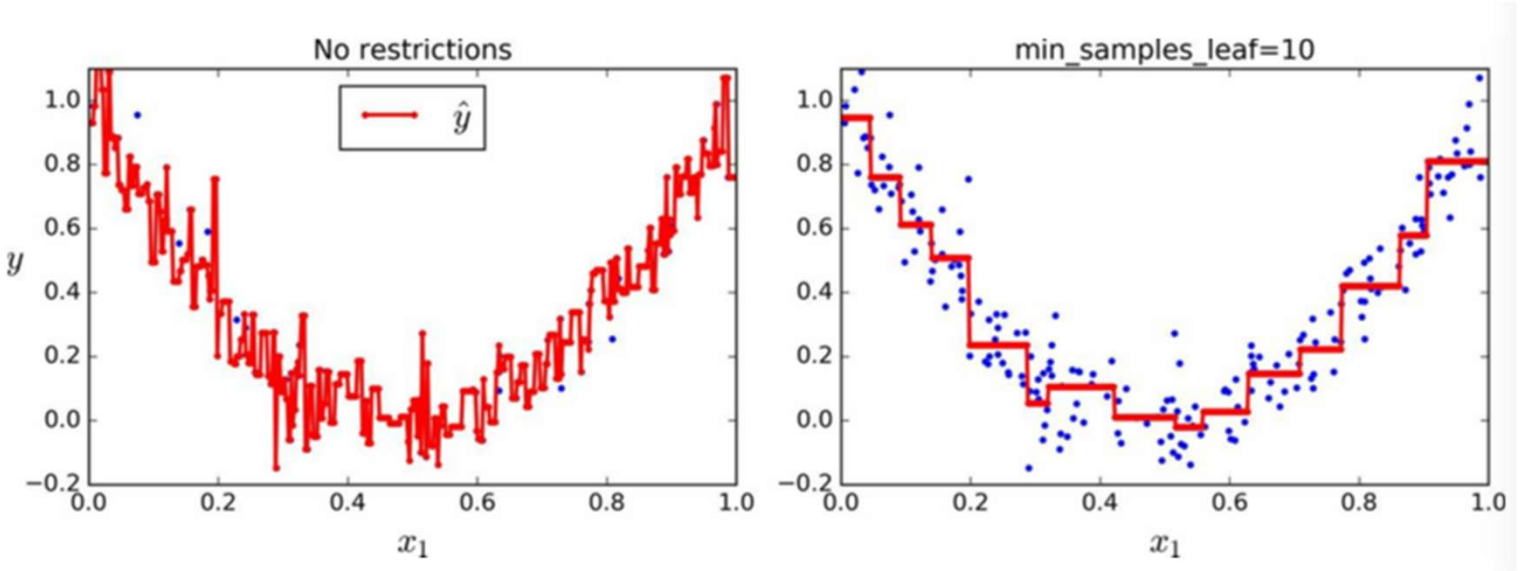


3. CART cost function

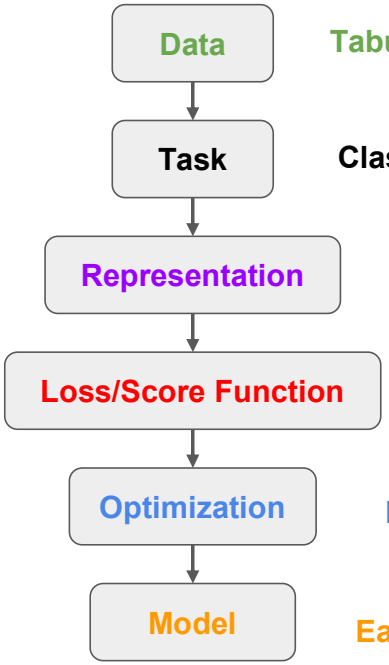
$$J(k, t_k) = \frac{m_{\text{left}}}{m}\text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m}\text{MSE}_{\text{right}}$$

$$\text{where} \begin{cases} \text{MSE}_{\text{node}} = \sum_{i \in \text{node}} (\hat{y}_{\text{node}} - y^{(i)})^2 \\ \hat{y}_{\text{node}} = \frac{1}{m_{\text{node}}} \sum_{i \in \text{node}} y^{(i)} \end{cases}$$

4. Regression Overfitting and Regularization



Recap: Decision Trees



Tabular form (both numerical and categorical data)

Classification , Regression

Tree ADT

Impurity / Entropy / Information Gain

Heuristic, Greedy

Easy to explain model of a Tree

Output:

- Regression
  Average of target value in the leaf node

- Classification:
  Majority class in the leaf node

Summary

+ Easy to understand (by human)
+ Computationally efficient
+ Handle both numerical and categorical data
+ Eager learning: do not have to carry training data around
+ Building block for various ensemble methods

- Heuristic training techniques
- Finding partition of space that minimizes error is NP-hard
- Use greedy approaches with limited theoretical work