Machine learning with Python

Decision Trees

Decision Trees

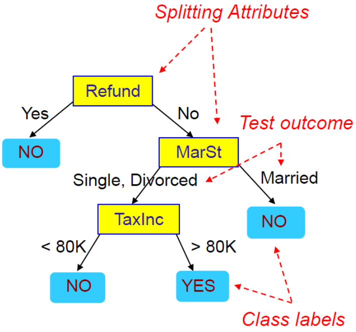
- Decision trees are a classification methodology, wherein the classification process is modeled with the use of a set of hierarchical decisions on the feature variables, arranged in a tree-like structure.
- The decision at a particular node of the tree is typically a condition on one or more feature variables(split criterion)
- The split criterion divides the training data into two or more parts
- Tree structure:
    - Root node - the topmost decision node
    - Internal node- consecutive decision node denote a test on an attribute
    - Branch - represents an outcome of the test
    - Leaf node - represents a classification label or decision

Decision Tree

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

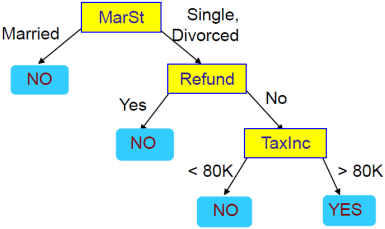
Training Data



Model: Decision Tree

Decision Tree

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree trained on the same dataset

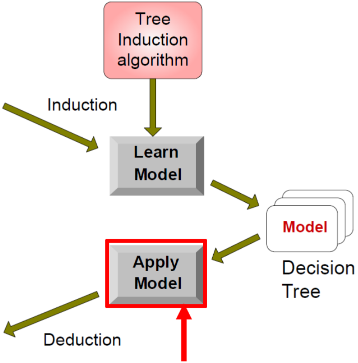
Decision Tree Classification

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

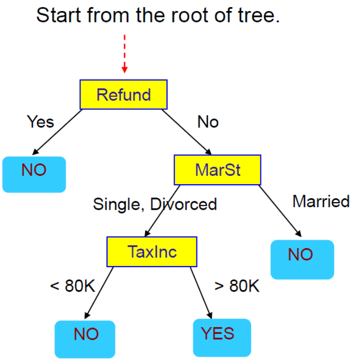
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Decision Tree Classification

Step 1

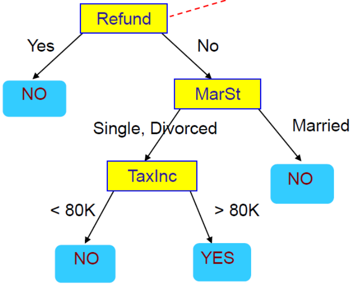


Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Decision Tree Classification

Step 2

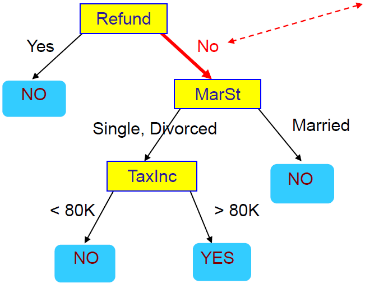


Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Decision Tree Classification

Step 3



Test Data

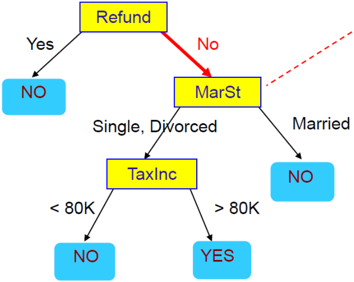
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Decision Tree Classification

Step 4

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

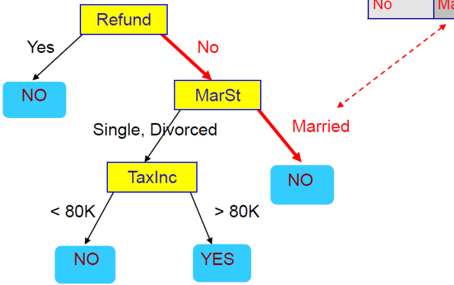


Decision Tree Classification

Step 5

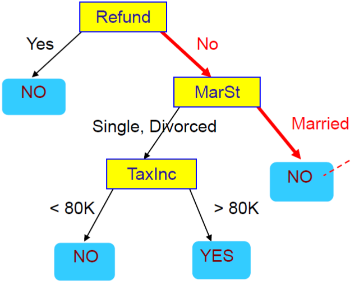
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Decision Tree Classification

Step 6



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

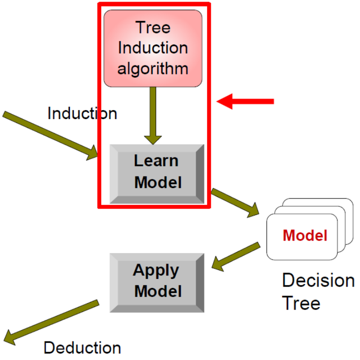
Divide and Conquer

- Decision trees are built using a heuristic called recursive partitioning.
- This approach splits the data into subsets, which are then split repeatedly into even smaller subsets
- The process stops when the algorithm determines the data within the subsets are sufficiently homogenous, or another stopping criterion has been met.
  - Start with bare root node that will grow into a mature tree
  - The decision tree algorithm must choose a feature to split upon
  - The examples are then partitioned into groups according to the distinct values forming branches
  - The algorithm continues to divide and conquer the data, choosing the best candidate feature each time to create another decision node
  - Algorithm stops when stopping criterion is reached

Tree Induction algorithm

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Generic algorithm

DecisionTree(Data Set: D)
begin

- Create root node containing D;

repeat

- Select an eligible node in the tree;
- Split the selected node into two or more nodes based on a pre-defined split criterion;

until no more eligible nodes for split;

- Prune overfitting nodes from tree;
- Label each leaf node with its dominant class;

end

Tree Induction Algorithms

- Hunt's Algorithm - one of the earliest
- CART
- ID3, C4.5 C5.0
- SLIQ
- SPRINT

Issues

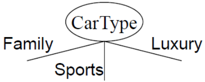
- How to split the records?
  - How to specify the attribute test condition?
  - How to choose the best split?
- When to stop splitting?
- How to classify a leaf node?

Test Condition

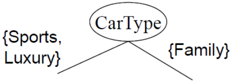
- Depends on two factors:
- Feature data type:
  - Nominal
  - Ordinal
  - Continuous
- Split pattern
  - 2-way split
  - multi-way split

Nominal features split

- Multi-way split- Number of partitons equal to the number of distinct values

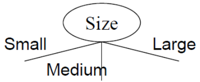


- Binary split - distinct values gruped in two partitions (optimize)

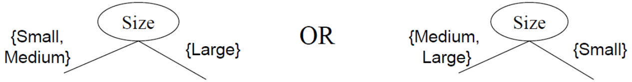


Ordinal features split

- Multi-way split - Number of partitons equal to the number of distinct values

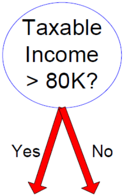


- Binary split - divides values into two subsets, order is maintained(optimize)

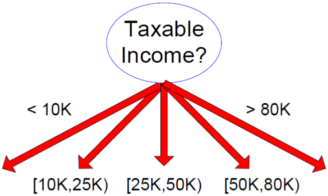


Continuous features split

- Multi-way split - discretization to ordinal categorical attribute
  - Static - intervals set at the beginning
  - Dynamic - interval bucketing, clustering
- Binary - all possibles splits are taken into account to find $(A < v)$ or $(A > v)$



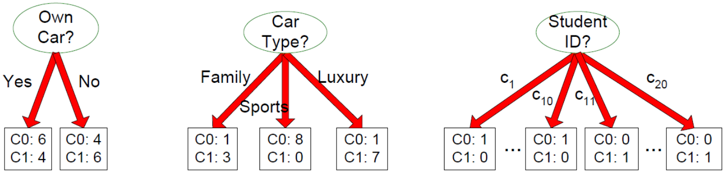
(i) Binary split



(ii) Multi-way split

How to determine the best split

- Which test condition is the best?



- We need to measure impurity

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, High degree of impurity	Homogeneous, Low degree of impurity

Choosing the best split

- The first challenge that a decision tree will face is to identify which feature to split
- The degree to which a subset of examples contains only a single class is known as purity
- There are various measurements of purity that can be used to identify the best decision tree splitting candidate:
- Entropy used in C5.0
- Gini used in ID3 and C4.5
- Classification error used in CART, SLIQ, SPRINT

The C5.0 algorithm

- One of the most well-known implementations is the C5.0 algorithm
- Developed by computer scientist J. Ross Quinlan as an improved version of his prior algorithm, C4.5
- C5.0 is a commercial algorithm but the source code for a single-threaded version of the algorithm was made publically available, and has been incorporated into programs such as R
- The C5.0 algorithm has become the industry standard to produce decision trees, because it does well for most types of problems directly out of the box.
- Performs nearly as well as Neural networks and SVM's but trees are much easier to understand and deploy

C5.0 - Strengths

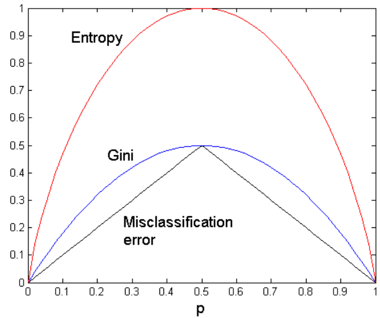
- An all-purpose classifier that does well on most problems
- Highly automatic learning process, which can handle numeric or nominal features, as well as missing data
- Excludes unimportant features
- Can be used on both small and large datasets
- Results in a model that can be interpreted without a mathematical background (for relatively small trees)
- More efficient than other complex models

C5.0 - Weaknesses

- Decision tree models are often biased toward splits on features having a large number of levels
- It is easy to overfit or underfit the model
- Can have trouble modeling some relationships due to reliance on axis-parallel splits
- Small changes in the training data can result in large changes to decision logic
- Large trees can be difficult to interpret and the decisions they make may seem counterintuitive

Splitting measures comparison

- All of the impurity measures take value zero (minimum) for the case of a pure node where a single value has probability 1
- A 50-50 split in a 2-class problem results in a maximum value of measures indicating maximum impurity.



Purity measures

$$Entropy(t) = -\sum_{i=1}^{c} p_i log_2 p_i \tag{1}$$

- For n classes, entropy ranges from 0 to log2(n)
- The minimum value indicates that the sample is completely homogenous, while the maximum value indicates that the data are as diverse as possible
- For a given segment of data (t), the term c refers to the number of class levels and pi refers to the proportion of values falling into class level i.

$$Gini(t) = 1 - \sum_{i=1}^{c} [p_i]^2 \tag{2}$$

$$Classerror(t) = 1 - max_i[p_i] \tag{3}$$

Example

C1	**0**
C2	**6**

$P(C1) = 0/6 = 0$ $P(C2) = 6/6 = 1$

Gini $= 1 – P(C1)^2 – P(C2)^2 = 1 – 0 – 1 = 0$

Entropy $= – 0 \log 0 – 1 \log 1 = – 0 – 0 = 0$

Error $= 1 – \max(0, 1) = 1 – 1 = 0$

C1	**1**
C2	**5**

$P(C1) = 1/6$ $P(C2) = 5/6$

Gini $= 1 – (1/6)^2 – (5/6)^2 = 0.278$

Entropy $= – (1/6) \log_2 (1/6) – (5/6) \log_2 (1/6) = 0.65$

Error $= 1 – \max(1/6, 5/6) = 1 – 5/6 = 1/6$

C1	**2**
C2	**4**

$P(C1) = 2/6$ $P(C2) = 4/6$

Gini $= 1 – (2/6)^2 – (4/6)^2 = 0.444$

Entropy $= – (2/6) \log_2 (2/6) – (4/6) \log_2 (4/6) = 0.92$

Error $= 1 – \max(2/6, 4/6) = 1 – 4/6 = 1/3$

Splitting issues

- Impurity measures favor attributes with large number of values
- A test condition with large number of outcomes may not be desirable
- To avoid that problem Information gain is adjusted by the entropy of the partition.
- Large number of partitions is penalized

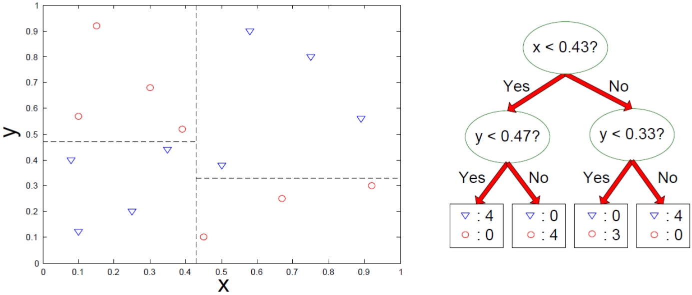
Stopping Criteria

- Algorithm stop expanding a node when all the records belong to the same class
- Algorithm stop expanding a node when all the records have similar feature values
- Early termination - pruning

Data fragmentation

- Data fragmentation is a problem in creating decision trees
- Number of instances gets smaller down the tree
- It may cause a problem to make any statistically significant decision if number of instances at the leaf nodes would be too small
- Solution to that problem is a lower bound on the number of items per leaf node in the stopping criterion

Axis-parallel splits



- Decision boundary is a border line between two neighboring regions of different classes
- It is parallel to axes because test condition involves a single attribute at-a-time

Pruning the decision tree

- A decision tree can continue to grow indefinitely until each example is perfectly classified or the algorithm runs out of features to split on
- If the tree grows overly large, will be overly specific and the model will be overfitted to the training data.
- The process of pruning a decision tree involves reducing its size such that it generalizes better to unseen data
- Two types of pruning:
  - pre-pruning
  - post-pruning

Pre-pruning

- Stop the tree from growing once it reaches a certain number of decisions or when the decision nodes contain only a small number of examples.
- Typical rules
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same
- Restructive rules
  - Stop if number of instances is less than user-specified threshold
  - Stop if expanding the current node does not improve impurity measures

Post-pruning

- Involves growing a tree that is intentionally too large and pruning leaf nodes to reduce the size of the tree to a more appropriate level.
- The nodes are trimmed in a bottom-up fashion
- If tree improves after trimming, replace sub-tree by a leaf node
- Class label of leaf node is determined from majority class of instances in the sub-tree
- This is often a more effective approach than pre-pruning