Decision Trees

Nishanth Nair

10 Nov,2019

Overview

- A decision tree represents a hierarchical segmentation of data.
- It is a supervised machine learning technique whose goal is to recursively partition the world into homogenous zones (i.e zones having constant target value).
- The original segement is called the root node. The root node is partitioned into two or more segments by applying a series of simple rules over input variables.
- The resulting segments are also called nodes,and the final segments are called leaf nodes or leaves.
- Nodes test the value of a feature.
- Leaf nodes output a predicted class or value.
- **Decision Boundaries**:
  - Decision trees limited to "hyper-rectangle" partitions of training data.
  - Can't form linear hypothesis
  - Can only approximate diagonal line as a step function.
  - Limitation not a problem since most often we work in high dimensional space, where the true function isn't a line.
- **Hypothesis space**:
  - If there are n binary variables, how many distinct decision trees are possible?
    - Equal to number of binary functions over n attributes
    - Equal to number of distinct truth tables with 2^n rows.
    - Equal to $(2)^{(2^n)}$
- **Building a tree**:
  - Goal 1: Build a tree consistent with training data.
    - Could choose any feature to be the root.
    - Would eventually have large tree where each leaf is a training example.
    - Would result in complex decision boundary.
  - Goal 2: Build a tree that generalizes to new data.
    - Can be achieved by applying Occam's Razor, keeping tree small.
    - Note: There will always be tradeoff between model complexity and ability to generalize.

- **Disadvantages**
  - Over fitting: Over fitting is one of the most practical difficulty for decision tree models. This problem gets solved by setting constraints on model parameters and pruning .
  - Not fit for continuous variables: While working with continuous numerical variables, decision tree loses information when it categorizes variables into different categories.

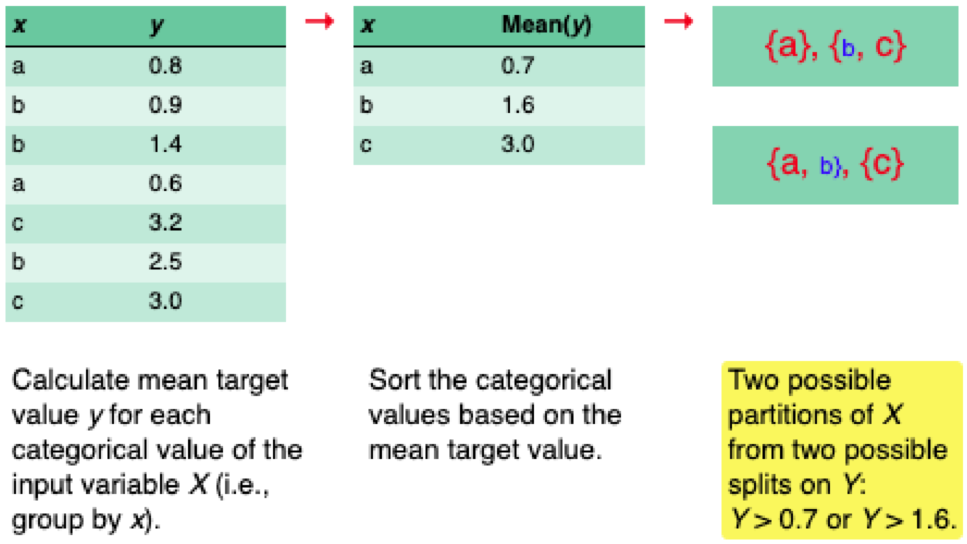
Regression Trees vs Classification Trees

- Regression trees are used when dependent variable is continuous. Classification trees are used when dependent variable is categorical.
- In case of regression tree, the value obtained by terminal nodes in the training data is the mean response of observation falling in that region.
- In case of classification tree, the value (class) obtained by terminal node in the training data is the mode of observations falling in that region. - Both the trees divide the predictor space (independent variables) into distinct and non-overlapping regions.
- Both the trees follow a top-down greedy approach known as recursive binary splitting.
  - Top Down because it begins from the top of tree when all the observations are available in a single region and successively splits the predictor space into two new branches down the tree.
  - Greedy because, the algorithm cares (looks for best variable available) about only the current split, and not about future splits which will lead to a better tree.
- This splitting process is continued until a user defined stopping criteria is reached. For example: we can tell the the algorithm to stop once the number of observations per node becomes less than 50.
- In both the cases, the splitting process results in fully grown trees until the stopping criteria is reached.
- But, the fully grown tree is likely to overfit data, leading to poor accuracy on unseen data. Pruning is one of the technique used tackle overfitting.

Data Pre-processing

1. Ordered categorical input attribute:
   - Check each split point to find best split.
   - For Regression tree's use variance
   - For classification tree's, use Info gain or Gini.
2. Un-ordered categorical input attribute:
   - Using Breiman's theorem:
     - Use Breiman's theorem for both classification and regression.
     - **Breiman's theorem**: For unordered domains, split predicates are of the form $X_i \in v_1, v_2, ..., v_k, where v_1, v_2, .., v_k \in P(D_(X_i)), the power set of D_(X_i).$

    – Breiman presents an algorithm for finding the best split predicate for a categorical attribute without evaluating all possible subsets of X_i. - The optimal split predicate is a subsequence in the list of values for X_i sorted by the average Y value.



x	*y*
a	0.8
b	0.9
b	1.4
a	0.6
c	3.2
b	2.5
c	3.0

x	Mean(*y*)
a	0.7
b	1.6
c	3.0

{a}, {b, c}

{a, b}, {c}

Calculate mean target value *y* for each categorical value of the input variable *X* (i.e., group by *x*).

Sort the categorical values based on the mean target value.

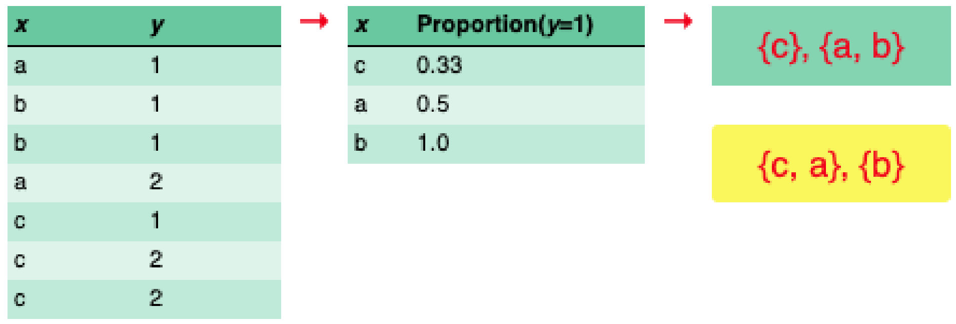
Two possible partitions of *X* from two possible splits on *Y*: *Y* > 0.7 or *Y* > 1.6.

Figure 1: Example using Breiman's theorem

- Binning Method:
    – Select candidate partition of input attribute based upon the ordered target variable.
    – Score candidate bin.
    – Pick bin based on these candidate partitions.

3. Unordered categorical input and output

- For a categorical feature with M possible values (categories), one could come up with 2M-1-1 split candidates.
- For binary classification and regression, we can reduce the number of split candidates to M - 1 by ordering the categorical feature values by the average label.
- Example: For a binary classification problem with one categorical feature with three categories, A, B, and C, whose corresponding proportions of label 1 are 0.5, 1.0, and 0.33:
    – The categorical features are ordered as C, A, B
    – The two split candidates are {C,A} | {B} and {C} | {A,B}, where | denotes the split.



x	*y*
a	1
b	1
b	1
a	2
c	1
c	2
c	2

x	Proportion(*y*=1)
c	0.33
a	0.5
b	1.0

Figure 2: Example using proportions

4. Categorical input and multi class output:
   - In a multiclass classification, all 2M-1-1 possible splits are used whenever possible.
   - If number of splits is greater than number of classes (or maxBins), a (heuristic) method similar to the method used for binary classification and regression is used.
   - The M categorical feature values are ordered by impurity, and the resulting M - 1 split candidates are considered.
5. Numeric attributes
   - These are discretized using the following approaches:
     - Percentile based bins
     - Equal size bin histograms
     - If data is too big, use **reservoir sampling**, which is used to downsample and allows to calculate histogram by samples.

Algorithm to build a tree

- Goal: Find a tree that is consistent with training examples.
- Strategy: Recursively choose most significant attribute as root of subtree

```
GrowTree (S):
  if stoppingCriteria($S_L$):
    return new Leaf(FindPrediction($S_L$))
  elif stoppingCriteria($S_R$):
    return new Leaf(FindPrediction($S_R$))
  else:
    $S_L, S_R$ = findbestSplit(S)
    return new node (GrowTree ($S_L$), GrowTree ($S_R$))
```

Find Best Split

- The decision of making strategic splits heavily affects a tree's accuracy.
- The decision criteria is different for classification and regression trees.
- The creation of sub-nodes increases the homogeneity (purity) of resultant sub-nodes.Decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.
- The algorithm selection is based on type of target variables.

1. Reduction in variance

- Used for continuous target variables (regression problems)
- The split with lower variance is selected as the criteria to split the population
- Which measure to use?

  We use summary statistics (mode,median,mean) to recursively partition data into zones. Assume that we have a list of numbers $(x_1, x_2, .., x_n)$ that we want to summarize in terms of central behavior. We want our summary to be a single number, s. We choose an s that has the smallest discrepancy or error between itself and each number in the list. Depending on the choice, we get the following loss functions:

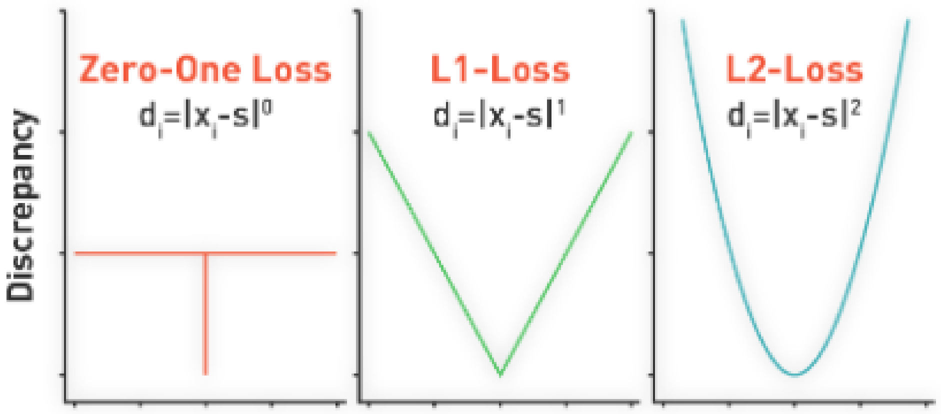


Figure 3: Loss Functions

1. Mode: Also called Zero-one loss, $d_i = |x_i - s|^0$
   - This says, discrepency is 0 if $x_i = s$ and 1 if $x_i \neq s$
2. Median: Also ccalled Absolute loss (L1 Loss), $d_i = |x_i - s|^1$
   - This says, discrepency is equal to the distance between x_i and s.
3. Mean: Squared error loss(L2 Loss) $d_i = |x_i - s|^2$
   - This says, discrepency is the squared distance between x_i and s.

In regression decision trees: - For centrality: Summarize zones using mean value. - For spread: The quality of the splits is characterized using the variance. (Purity)

$$Variance = \frac{\sum(X_i - \bar{X})^2}{n}, \text{(Using L2 Loss)}$$

- **Steps:**
  - Compute variance of training data as root.
  - Calculate variance for each attribute.
  - Calculate variance for each split as weighted average of each attribute variance.
  - Choose attribute whose split has lowest variance.
- Example:

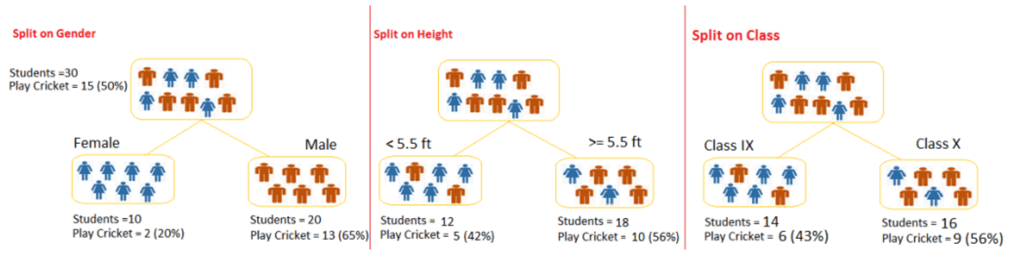


Figure 4: Example

1. Variance for root node:
   - Mean $= \frac{15*1+15*0}{30} = 0.5$
   - Variance $= \frac{(x_i-0.5)^2}{30} = \frac{15*0.5^2+15*0.5^2}{30} = 0.5^2 = 0.25$
2. Compute variance of gender attribute:
   - Mean for Female value $= \frac{2*1+8*0}{10} = 0.2 \implies$ Variance $= \frac{2*(1-0.2)^2+8*(0-0.2)^2}{10} =$ 0.16
   - Mean for male value = 0.65 and variance = 0.23
   - Variance for split = Weighted variance of values $= \frac{10}{30}.0.16 + \frac{20}{30}.0.23 = 0.21$
3. Compute variance for class attribute:
   - Mean for Class IX value = 0.43 and variance = 0.24
   - Mean for Class X value = 0.56 and variance = 0.25
   - Variance for split = Weighted variance of values $= \frac{14}{30}.0.24 + \frac{16}{30}.0.25 = 0.25$

Since variance for gender is lowest, this will have maximum purity and thus, first split will be based on gender.

2. Information Gain

- Used with categorical target variable (Classification problems)
- Choose the split which has lowest entropy compared to parent node and other splits.
- **Entropy:**

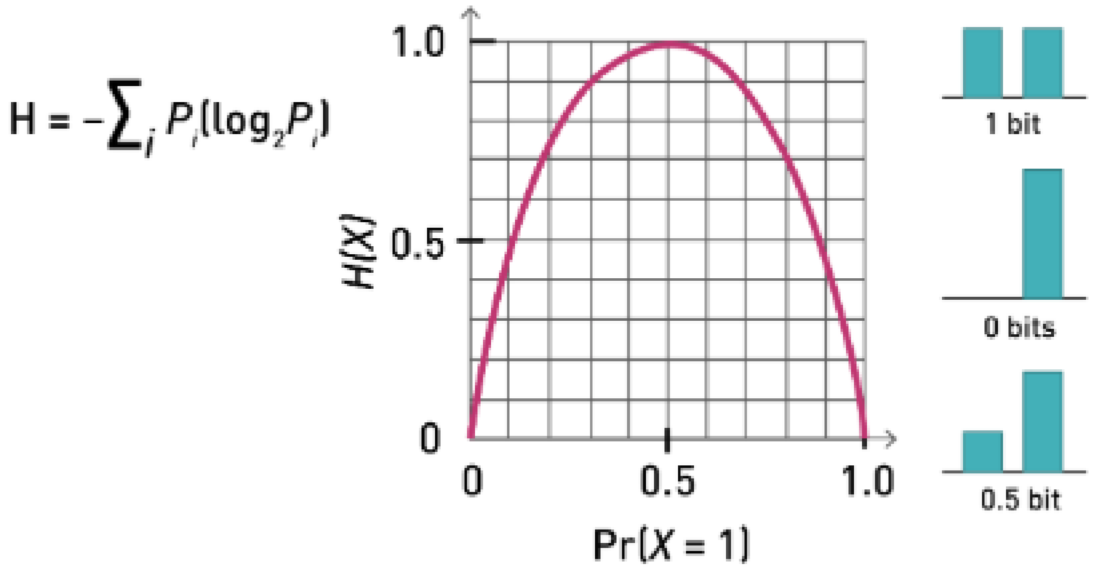


Figure 5: Entropy Function

  - Measure of uncertainty
  - The more uncertainty there is about a random variable, the more information is conveyed by the value
  - Maximized when there is complete uncertainty through uniform distribution
  - 0 when there is complete certainty

- **Information Gain** : Describes a change in entropy
  - Difference between the entropy before and after change
  - A.K.A Kullback-Leibler divergence or Relative entropy
  - Try to maximize information gain$

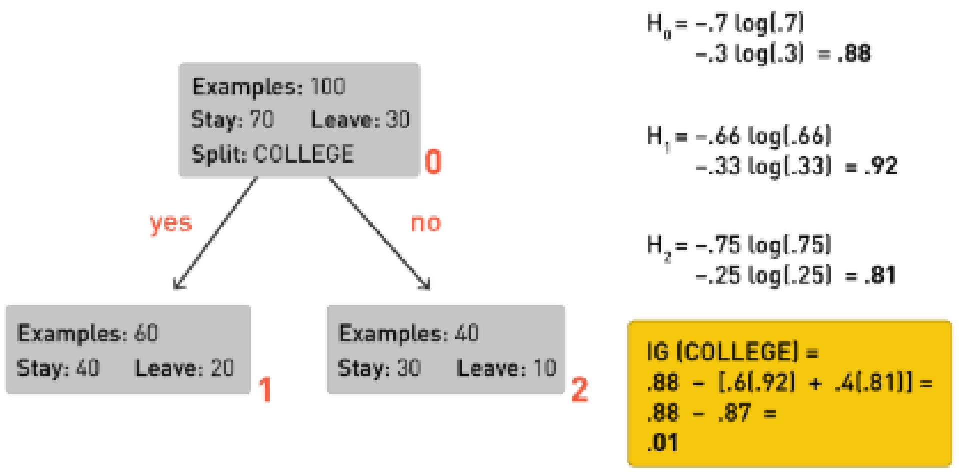

Examples: 100
Stay: 70 Leave: 30
Split: COLLEGE
0
yes
no
Examples: 60
Stay: 40 Leave: 20
1
Examples: 40
Stay: 30 Leave: 10
2
H0 = -.7 log(.7) -.3 log(.3) = .88
H1 = -.66 log(.66) -.33 log(.33) = .92
H2 = -.75 log(.75) -.25 log(.25) = .81
IG (COLLEGE) = .88 - [.6(.92) + .4(.81)] = .88 - .87 = .01

Figure 6: Information Gain Example

- **Steps:**
  - Compute entropy of training data as root.
  - Calculate entropy for each attribute:
    - Calculate entropy of each value category of attribute.
    - Calculate split entropy = weighted entropy across categories
  - Choose attribute whose split has lowest entropy.
- Example:

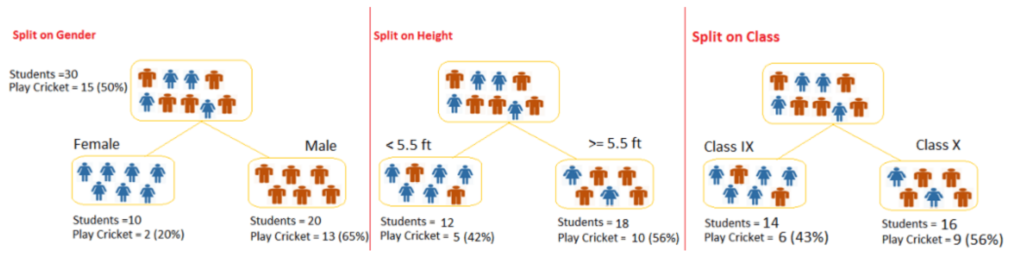


Figure 7: Example

1. Entropy for root node $= -\frac{15}{30}.\log_2(\frac{15}{30}) - \frac{15}{30}.\log_2(\frac{15}{30}) = 1$.Here 1 shows that it is a impure node.
2. Compute entropy of gender attribute:
   - Entropy for Female value $= -\frac{2}{10}.\log_2(\frac{2}{10}) - \frac{8}{10}.\log_2(\frac{8}{10}) = 0.72$
   - Entropy for male value, $-\frac{13}{20}.\log_2(\frac{13}{20}) - \frac{7}{20}.\log_2(\frac{7}{20}) = 0.93$
   - Entropy for split = Weighted entropy of sub-nodes $= \frac{10}{30}.0.72 + \frac{20}{30}.0.93 = 0.86$
3. Compute entropy for class attribute:
   - Entropy for Class IX value, $-\frac{6}{14}.\log_2(\frac{6}{14}) - \frac{8}{14}.\log_2(\frac{8}{14}) = 0.99$
   - Entropy for Class X value, $-\frac{9}{16}.\log_2(\frac{9}{16}) - \frac{7}{16}.\log_2(\frac{7}{16}) = 0.99$.
   - Entropy for split $= \frac{14}{30}.0.99 + \frac{16}{30}.0.99 = 0.99$

Since entropy for gender is lowest, this will have maximum information gain (i.e $1 - 0.86 = 0.14$) and thus, first split will be based on gender.

3. Chi-Square

TBD

4. Gini

TBD

Stopping Criteria

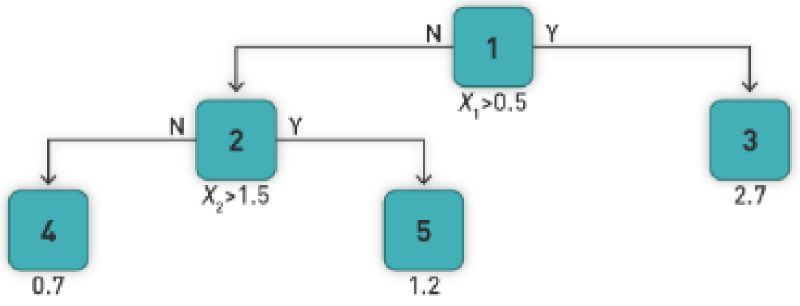
- Many different heuristics:
  - When leaf is pure, i.e target variable does not vary too much: $Var(y_i) < \varepsilon$
  - When the number of examples in the leaf is too small
  - When uncertainty in the node is low

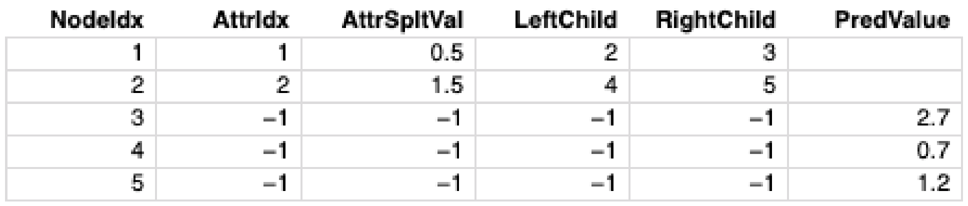
Find Prediction

- Regression (and binomial logistic regression)
  - Predict average y_i of the examples in the leaf.
  - Build a linear regression model on the examples in the leaf.
- Classification
  - Predict most common y_i of the examples in the leaf.

Representation of a decision tree

Most common representation is in the form of a table making it easier to persist,pass around and lookup.





NodeIdx	AttrIdx	AttrSpltVal	LeftChild	RightChild	PredValue
1	1	0.5	2	3	
2	2	1.5	4	5	
3	−1	−1	−1	−1	2.7
4	−1	−1	−1	−1	0.7
5	−1	−1	−1	−1	1.2

Figure 8: Example of Decision tree representation

How to predict using the above representation:

```
input: X, Tree
output: Predicted value

index = rootNodeIdx
while(Tree[index].Attridx != −1):
    value = X[Tree[index].Attridx]
    split = Tree[index].AttrSplitVal
    if(value > split):
        index = Tree[index].RightChild
    else:
        index = Tree[index].LeftChild

 return Tree[index].PredValue
```

Avoiding Over-fitting

Overfitting is one of the key challenges faced while modeling decision trees. If there is no limit set of a decision tree, it will give you 100% accuracy on training set because in the worse case it will end up making 1 leaf for each observation. Thus, preventing overfitting is pivotal while modeling a decision tree and it can be done in 2 ways:

1. Setting constraints on tree size

2. Tree pruning

1. Setting constraints on tree size

- Minimum samples for a node split
  - Defines the minimum number of samples (or observations) which are required in a node to be considered for splitting.
  - Used to control over-fitting.
  - Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.
  - Very high values can lead to under-fitting hence, it should be tuned using CV.
- Minimum samples for a terminal node (leaf)
  - Defines the minimum samples (or observations) required in a terminal node or leaf.
  - Used to control over-fitting similar to min_samples_split.
  - Generally lower values should be chosen for imbalanced class problems because the regions in which the minority class will be in majority will be very small.
- Maximum depth of tree (vertical depth)
  - The maximum depth of a tree.
  - Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
  - Should be tuned using CV.
- Maximum number of terminal nodes
  - The maximum number of terminal nodes or leaves in a tree.
  - Can be defined in place of max_depth. Since binary trees are created, a depth of 'n' would produce a maximum of 2^n leaves.
- Maximum features to consider for split
  - The number of features to consider while searching for a best split. These will be randomly selected.
  - As a thumb-rule, square root of the total number of features works great but we should check upto 30-40% of the total number of features.
  - Higher values can lead to over-fitting but depends on case to case.

2. Pruning

The technique of setting constraint is a greedy-approach. In other words, it will check for the best split instantaneously and move forward until one of the specified stopping condition is reached.A decision tree with constraints won't see available options ahead and adopt a greedy approach.On the other hand if we use pruning, we in effect look at a few steps ahead and then make a choice getting better results.

In this approach, we compute a significance value (pchance) at each split using chi-squared test.

Proccess is as follows:

- Build full decision tree.
- Start with deepest nodes, deleting splits where pchance > maxpchance.
- Work upward until no remaining prunable nodes.
- Form of regularization:
  - Maxpchance is a regularization parameter.
  - Estimate by using development data.
  - Hyperparameter of training routine.

Ensemble Methods

- Ensemble methods involve group of predictive models to achieve a better accuracy and model stability.
- Ensemble learning is one way to execute the bias variance trade-off analysis.
  - Bias means, 'how much on an average are the predicted values different from the actual value.'
  - Variance means, 'how different will the predictions of the model be at the same point if different samples are taken from the same population'.
- Some ensemble methods are:
  - Bagging
  - Boosting

1. Bagging

Bagging is a technique used to reduce the variance of our predictions by combining the result of multiple classifiers modeled on different sub-samples of the same data set.

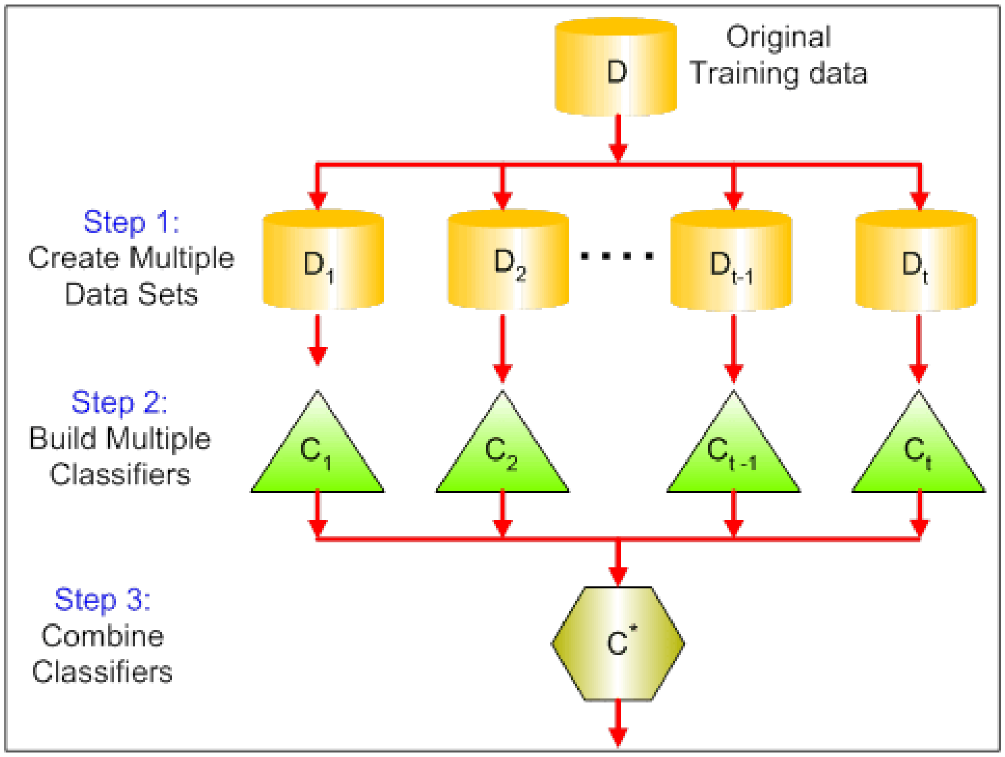


Figure 9: Bagging

The process is as follows:

1. Create Multiple DataSets (aka **bootstrap sampling**):
    - Sampling is done with replacement on the original data and new datasets are formed.
    - The new data sets can have a fraction of the columns as well as rows, which are generally hyper-parameters in a bagging model
    - Taking row and column fractions less than 1 helps in making robust models, less prone to overfitting
2. Build Multiple Classifiers:
    - Classifiers are built on each data set.
    - Generally the same classifier is modeled on each data set and predictions are made.
3. Combine Classifiers:
    - The predictions of all the classifiers are combined using a mean, median or mode value depending on the problem at hand.
    - The combined values are generally more robust than a single model.

Example: **Random Forest**

- Generate artificial training set through bootstrapping with replacement.
- Build decision tree.
- Randomly choose subset of features; consider those as split points.
- Each tree is grown to the largest extent possible and there is no pruning.

- Repeat process to create multiple trees.
- Run test case through all trees.
- Predict by taking vote among trees.
  - majority votes for classification,
  - average for regression

One of benefits of Random forest is that it can handle thousands of input variables and identify most significant variables so it is considered as one of the dimensionality reduction methods.

2. Boosting

- Train sequence of models where each one emphasizes examples misclassified by previous model.
- AdaBoost:
  - Developed by Yoav Freund and Robert Schapire in early 2000s
  - General scheme for combining classifiers that together tend to reduce overfitting.
  - Idea is to maintain a weight or importance of each training example
- AdaBoost Algorithm:
  1. Set weight for each training example = $\frac{1}{n}$
  2. Train a classifier where objective respects the weights.
     - Run classifier over training examples.
  3. Reduce weights for correct examples; increase weights for misclassified examples.
  4. Return to step 2.
     - Second classifier is trained with objective that respects importance weights placed on each feature.