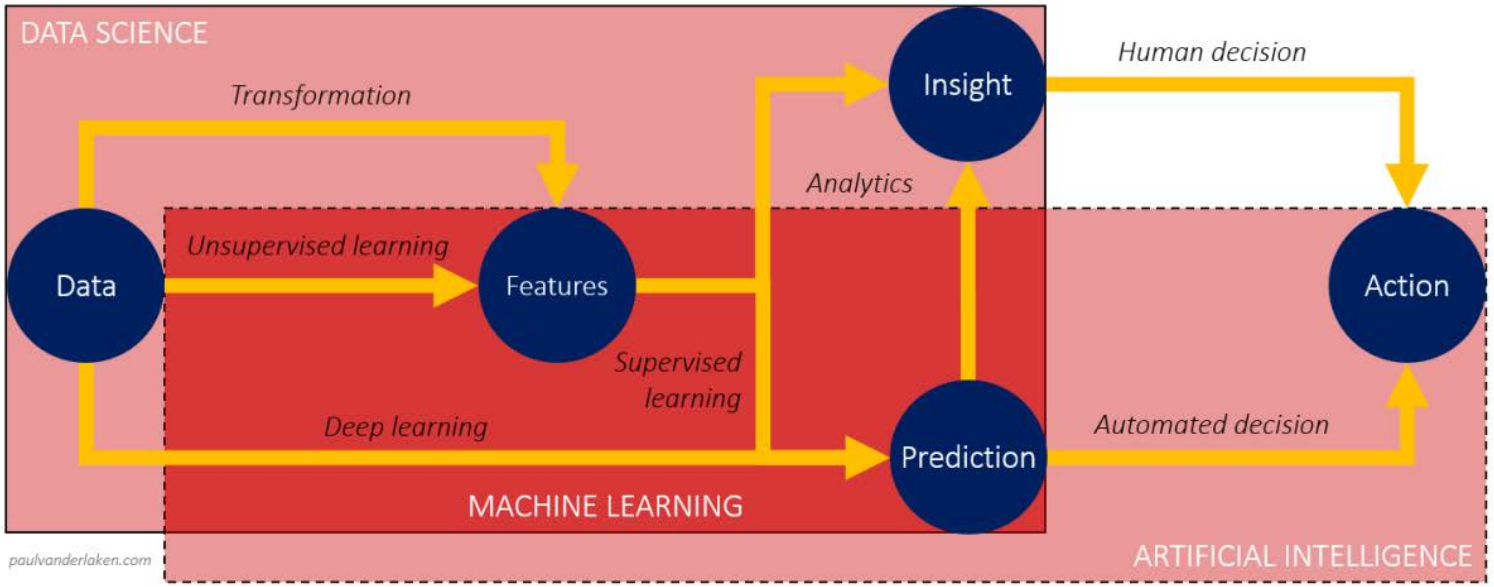
Tree-Based Methods



Tree-Based Methods

- *Tree-based* methods for regression and classification involve stratifying or segmenting the predictor space into a number of simple regions.

- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of machine learning approaches are known as *tree-based* methods.

- The basic idea of these methods is to partition the space and identify some representative centroids.

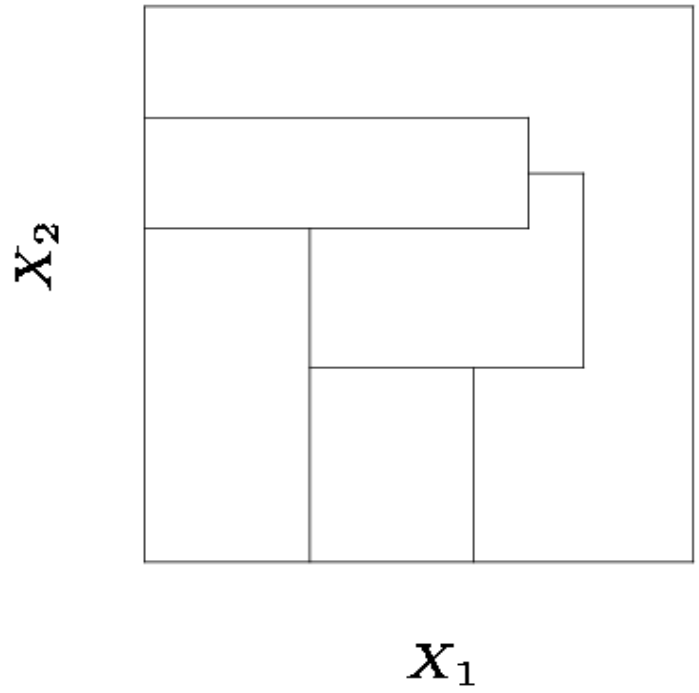
Regression Trees

- One way to make predictions in a regression problem is to divide the predictor space (i.e., all the possible values for X_1, X_2,…, X_p) into distinct regions, say R_1, R_2,…,R_k (*terminal nodes* or *leaves*).
- Then, for every X that falls into a particular region (say R_j) we make the same prediction.
- Regression trees are typically drawn *upside down*, in the sense that the leaves are at the bottom of the tree.
- The points along the tree where the predictor space is split are referred to as *internal nodes*. The segments of the trees that connect the nodes are *branches*.

Regression Trees (cont.)

- Suppose we have two regions R_1 and R_2 with $\hat{Y}_1 = 10$ and $\hat{Y}_2 = 20$, respectively.

- For any value of *X* such that $X \in R_1$, we would predict 10. For any value of *X* such that $X \in R_2$, we would predict 20.

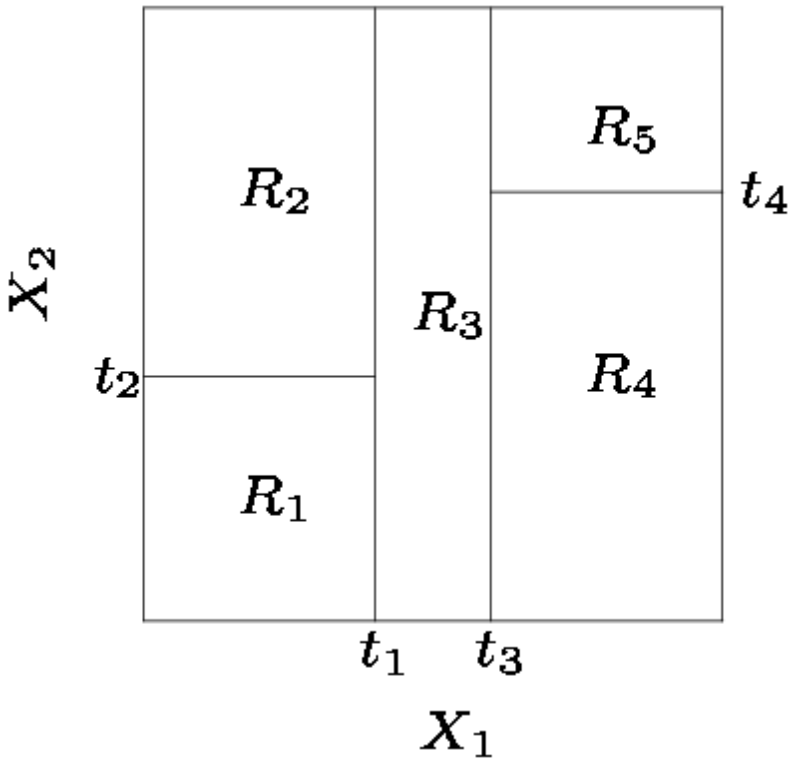


- In this illustrative example, we have two predictors and five distinct regions. Depending on which region the new *X* comes from, we would make one of five possible predictions for *Y*.

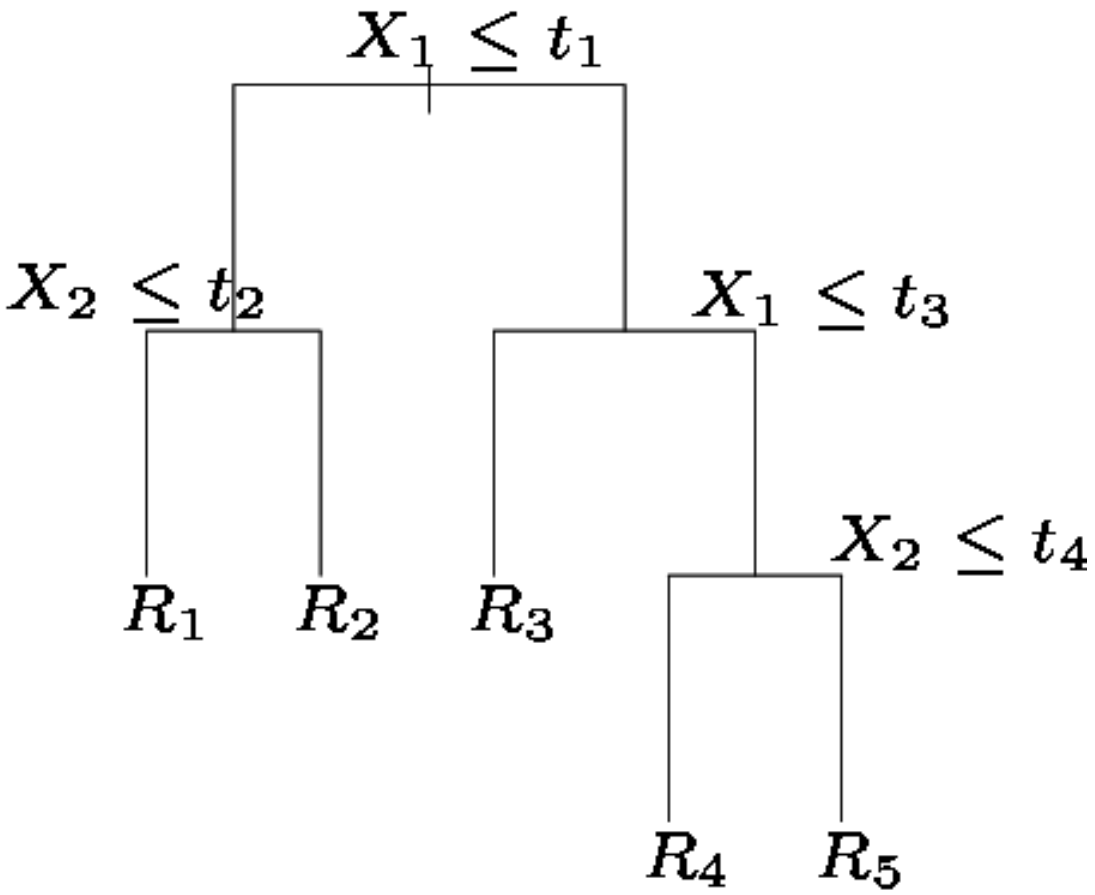
Regression Trees (cont.)

- Generally, we create the partitions by iteratively splitting one of the *X* variables into two regions.

- First split on $X_1 = t_1$
- If $X_1 < t_1$, split on $X_2 = t_2$
- If $X_1 > t_1$, split on $X_2 = t_4$
- If $X_1 > t_3$, split on $X_2 = t_4$



Regression Trees (cont.)



- When we create partitions this way, we can always represent them using a tree structure.
- As a result, this provides a very simple way to explain the model to a non-expert.

Classification Trees

- Classification trees are a hierarchical way of partitioning the space.
- We start with the entire space and recursively divide it into smaller regions.
- At the end, every region is assigned with a class label.
- We start with a medical example to get a rough idea about classification trees.

Classification Trees (cont.)

- One big advantage for classification trees is that the classifier generated is highly interpretable. For physicians, this is an especially desirable feature.

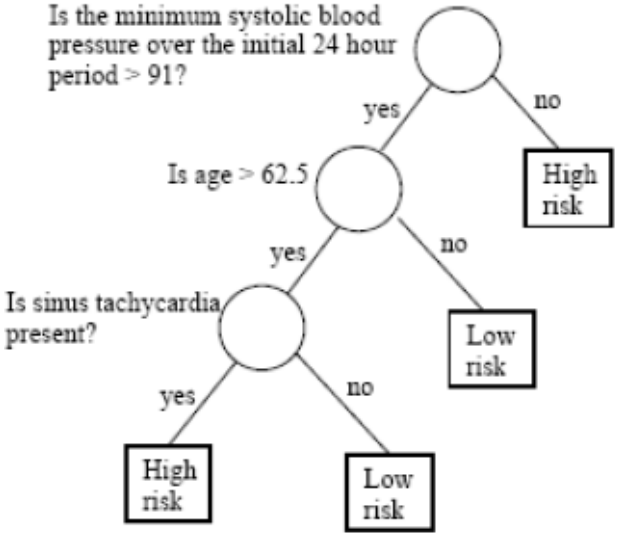
- In this example, patients are classified into one of two classes: high risk versus low risk.

- It is predicted that the high risk patients would not survive at least 30 days based on the initial 24-hour data.

- There are 19 measurements taken from each patient during the first 24 hours. These include blood pressure, age, etc.

Classification Trees (cont.)

- Here, we generate a tree-structured classification rule, which can be interpreted as follows:



- Only three measurements are looked at by this classifier. For some patients, only one measurement determines the final result.
- Classification trees operate similarly to a doctor's examination.

Classification Trees (cont.)

Notation:

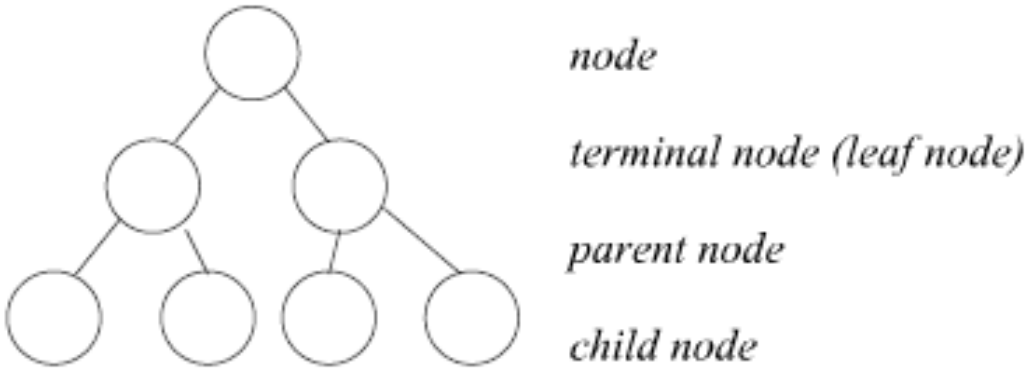
- The input vector is indicated by $X \in \boldsymbol{X}$ contains *p* features $X_1,\ldots,X_p$
- Tree structured classifiers are constructed by repeated splits of the space X into smaller and smaller subsets, beginning with **X** itself.
- *Node*: Any of the nodes in the classification tree, where each corresponds to a region in the original space.
- *Terminal Node*: The final node resulting from successive splits, where each is assigned a unique class.

Classification Trees (cont.)

Notation (cont.):

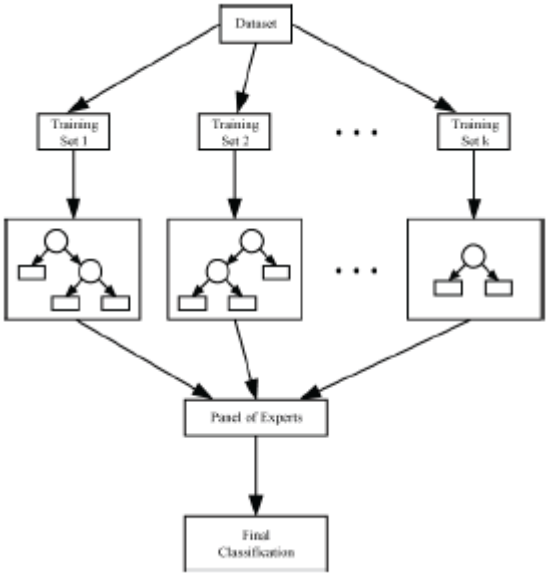
- *Parent Node*: those nodes that are split into two child nodes.

- *Child Node*: these result from the splitting of a parent node. Two child nodes are two different regions. Together they occupy the same region of the parent node.



Bagging

- There is a very powerful idea in the use of subsamples of the data and in averaging over subsamples through bootstrapping.
- Bagging exploits that idea to address the overfitting issue in a more fundamental manner.
- It was invented by Leo Breiman, who called it "**b**ootstrap **agg**regat**ing**" or simply "bagging.“
- In a classification tree, bagging takes a majority vote from classifiers trained on bootstrap samples of the training data.



Bagging (cont.)

- **Algorithm:** Consider the following steps in a fitting algorithm with a dataset having *N* observations and a binary response variable:
  1. Take a random sample of size *N with replacement* from the data (a bootstrap sample).
  2. Construct a “full” classification tree as usual but do not prune.
  3. Assign a class to each terminal node, and store the class attached to each case coupled with the predictor values for each observation.
  4. Repeat Steps 1-3 a large number of times.
  5. For each observation in the dataset, count the number of trees that it is classified in one category over the number of trees.
  6. Assign each observation to a final category by a majority vote over the set of trees. Thus, if 51% of the time over a large number of trees a given observation is classified as a "1'', that becomes its classification.

Bagging (cont.)

- Although there remain some important variations and details to consider, these are the key steps to producing "bagged'' classification trees.

- The idea of classifying by averaging over the results from a large number of bootstrap samples generalizes easily to a wide variety of classifiers beyond classification trees.

Random Forests

- In bagging, simply re-running the same machine learning algorithm on different subsets of the data can result in highly correlated predictors, which makes the model biased.

- On the other hand, random forests try to decorrelate the base learners by learning trees based on a randomly chosen subset of input variables, as well as a randomly chosen subset of data samples.

- Such models often have *very good predictive accuracy* and have been *widely used* in many applications.

Random Forests (cont.)

- Bagging constructs a large number of trees with bootstrap samples from a dataset.
- But now, as each tree is constructed, take a random sample of predictors before each node is split.
- For example, if there are twenty predictors, choose a random five as candidates for constructing the best split.
- Repeat this process for each node until the tree is large enough. And as in bagging, do not prune.

Random Forests (cont.)

- **Algorithm:** The random forests algorithm is very much like the bagging algorithm. Let *N* be the number of observations and assume for now that the response variable is binary:
  1. Take a random sample of size *N* with replacement from the data (bootstrap sample).
  2. Take a random sample without replacement of the predictors.
  3. Construct a split by using predictors selected in Step 2.
  4. Repeat Steps 2 and 3 for each subsequent split until the tree is as large as desired. Do not prune. Each tree is produced from a random sample of cases, and at each split a random sample of predictors.
  5. Drop the out-of-bag data down the tree. Store the class assigned to each observation along with each observation's predictor values.
  6. Repeat Steps 1-5 a large number of times (e.g., 500).
  7. For each observation in the dataset, count the number of trees that it is classified in one category over the number of trees.
  8. Assign each observation to a final category by a majority vote over the set of trees. Thus, if 51% of the time over a large number of trees a given observation is classified as a "1", that becomes its classification.

Random Forests (cont.)

- Random forests are among the very best classifiers invented to date!

- Random forests include three main tuning parameters.
  - **Node Size:** unlike in regression/classification trees, the number of observations in the terminal nodes of each tree of the forest can be very small. The goal is to grow trees with as little bias as possible.

  - **Number of Trees:** in practice, 500 trees is often a good choice.

  - **Number of Predictors Sampled:** the number of predictors sampled at each split would seem to be a key tuning parameter that should affect how well random forests perform. Sampling 2-5 each time is often adequate.

Random Forests (cont.)

- With classification accuracy as a criterion, bagging is in principle an improvement over classification/regression trees.
- It constructs a large number of trees with bootstrap samples from a dataset.
- Random forests is in principle an improvement over bagging.
- It draws a random sample of predictors to define each split.

Boosting

- Boosting, like bagging, is another general approach for improving prediction results for various machine learning methods.
- It is also particularly well suited to classification/regression trees.
- Rather than resampling the data, however, boosting weights on some examples during learning.
- Specifically, in boosting the trees are grown *sequentially*: each tree is grown using information from previously grown trees.

Boosting (cont.)

General Boosting Algorithm for Regression Trees:

- Unlike fitting a single large regression tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly.

- Given the current model, we fit a regression tree to the residuals from the model. We then add this new regression tree into the fitted function in order to update the residuals.

- Each of these trees can be rather small, with just a few terminal nodes, determined by a parameter in the algorithm.

- By fitting small trees to the residuals, we slowly improve in areas where it does not perform well.

Boosting (cont.)

Boosting for Classification

- Boosting iteratively learns weak classifiers; a weak classifier is one whose error rate is only slightly better than random guessing.

- The purpose of boosting is to sequentially apply the weak classification algorithm to repeatedly modified versions of the data, thereby producing a sequence of weak classifiers.

- The predictions from all of them are then combined through a weighted majority vote to produce the final prediction.

- Thus, the final result is the weighted sum of the results of weak classifiers.

Boosting (cont.)

- Boosting is remarkably resistant to overfitting, and it is fast and simple.
- In fact, it can often continue to improve even when the training error has gone to zero.
- It improves the performance of many kinds of machine learning algorithms.
- Boosting does not work when:
  - Not enough data, base learner is too weak or too strong, and/or susceptible to noisy data.

QUESTIONS????