Decision Tree

CS534 - Machine Learning

Yubin Park, PhD

Suppose we have a dataset

that has patients' medical notes

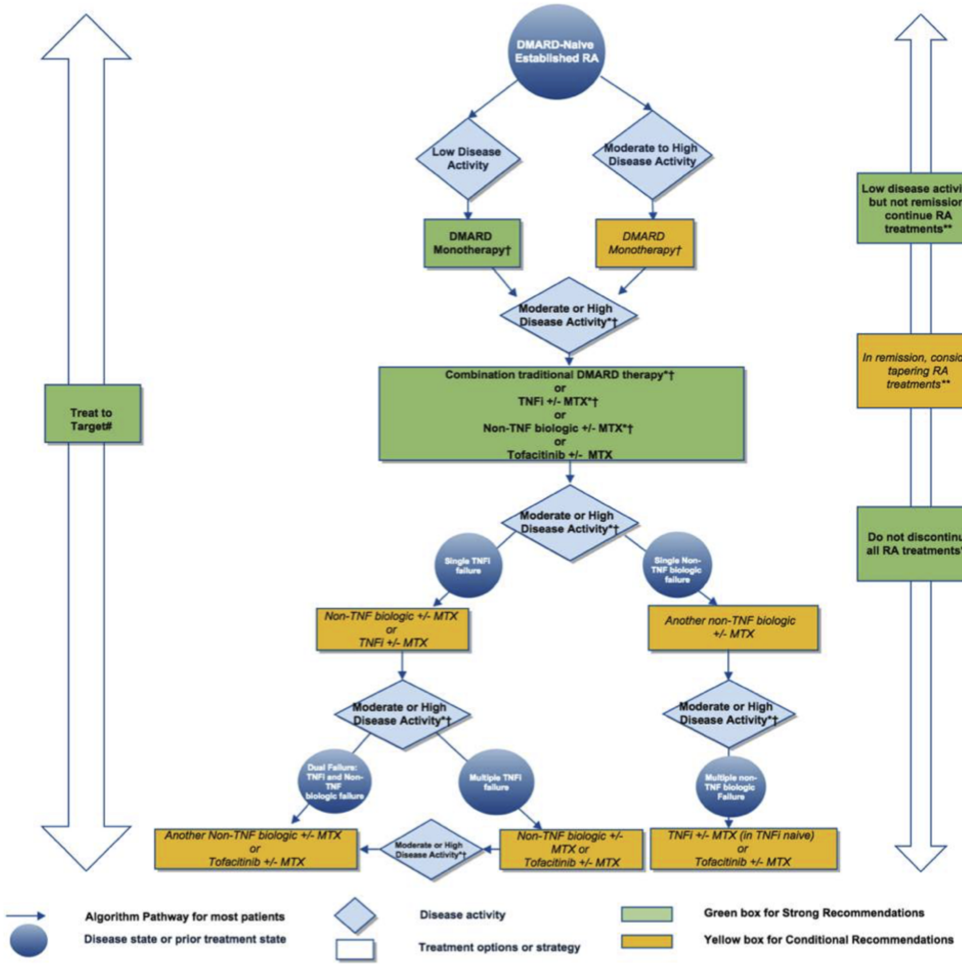
and treatment records.

We want to understand

the relationship between

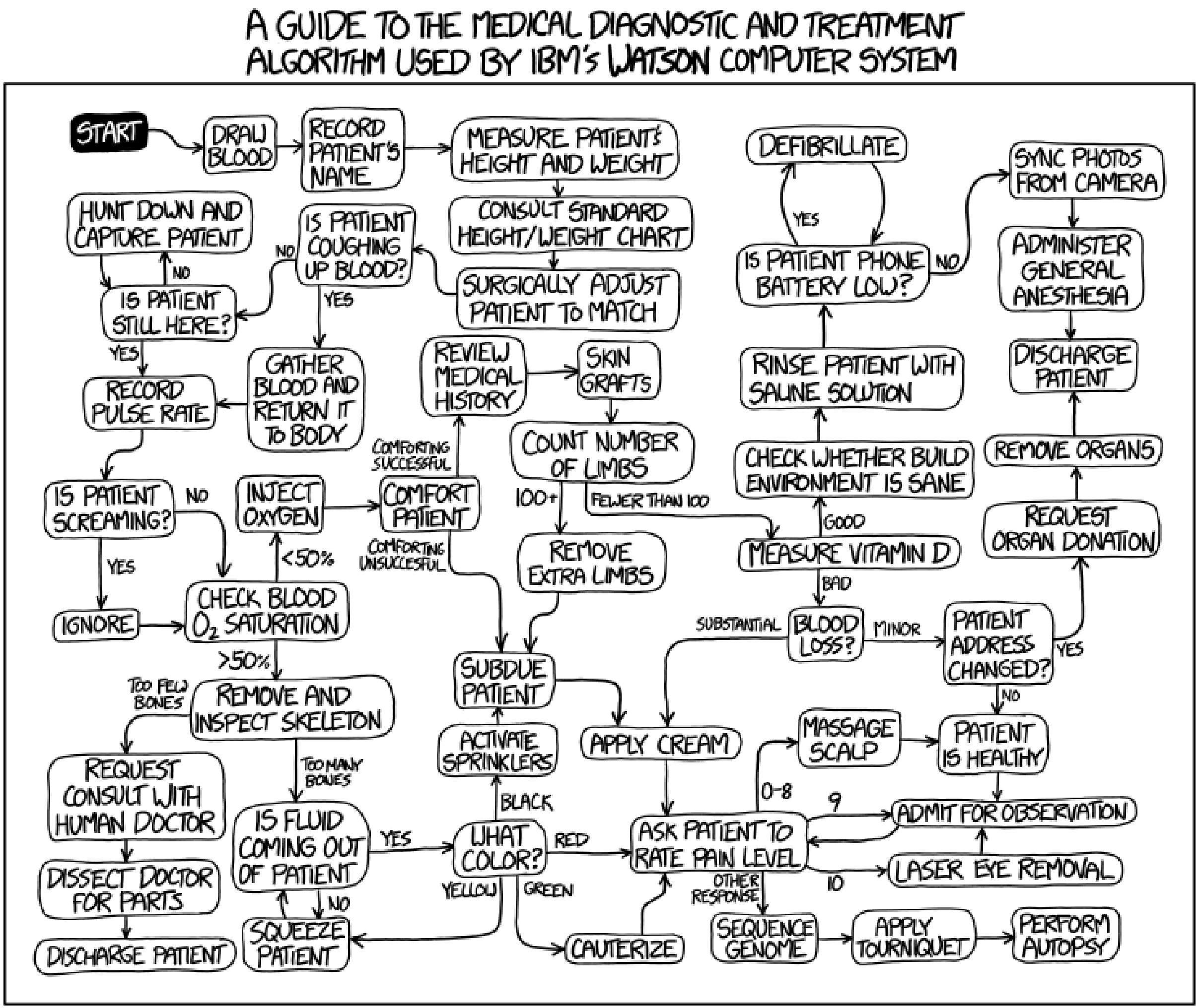
patients' conditions and their treatments.

So, how do doctors think?



https://www.rheumatology.org/Portals/0/Files/ACR%202015%20RA%20Guideline.pdf

Just for fun... https://xkcd.com/1619/



Such if-else rules may not be

easily captured in linear models.

Enter Decision Tree.

Beside modeling such if-else rules,

Decision Tree has many other advantages.

TABLE 10.1. *Some characteristics of different learning methods. Key:* ▲ *= good,* ◆ *=fair, and* ▼ *=poor.*

Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of "mixed" type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large N)	▼	▼	▲	▲	▼
Ability to deal with irrelevant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲

Chapter 10 of https://web.stanford.edu/~hastie/ElemStatLearn/

Decision Tree

Decision Tree partitions data with if-else rules and assigns constant predictive values to those partitions. Mathematically,

$$f(\mathbf{x}) = \sum_k c_k \boldsymbol{I}(\mathbf{x} \in \boldsymbol{R}_k)$$

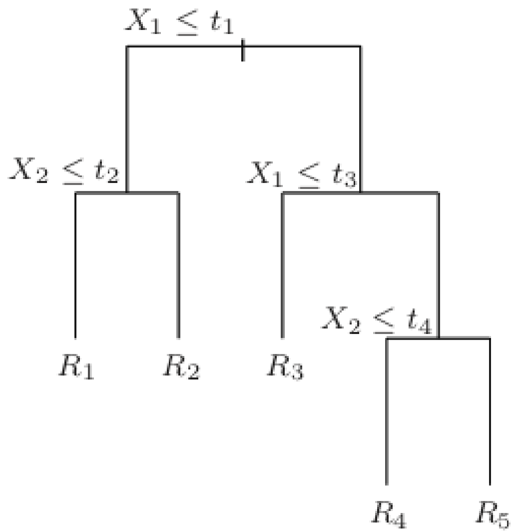
where $\boldsymbol{R}_k$ represents the kth partitioned region and c_k is the predictive value for the region.

Finding the optimal partitions is NP-hard.

Thus, we use a greedy approach. e.g. partitioning data one by one.

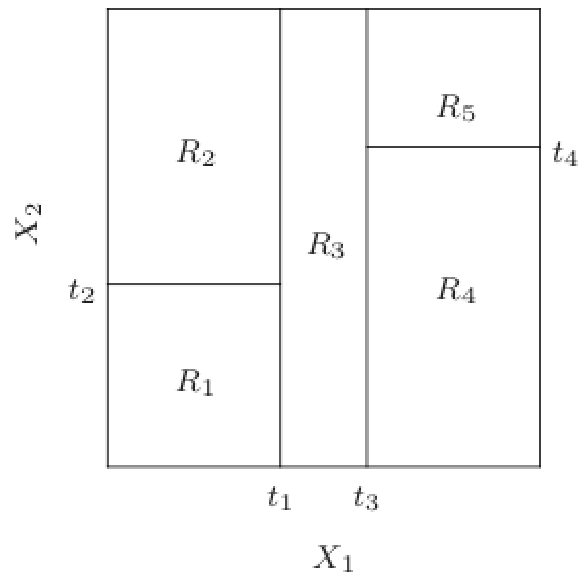
Decision Tree is also known as Recursive Partitioning.

Tree Representation



Chapter 9 of https://web.stanford.edu/~hastie/ElemStatLearn/

Partitioning Example



Chapter 9 of https://web.stanford.edu/~hastie/ElemStatLearn/

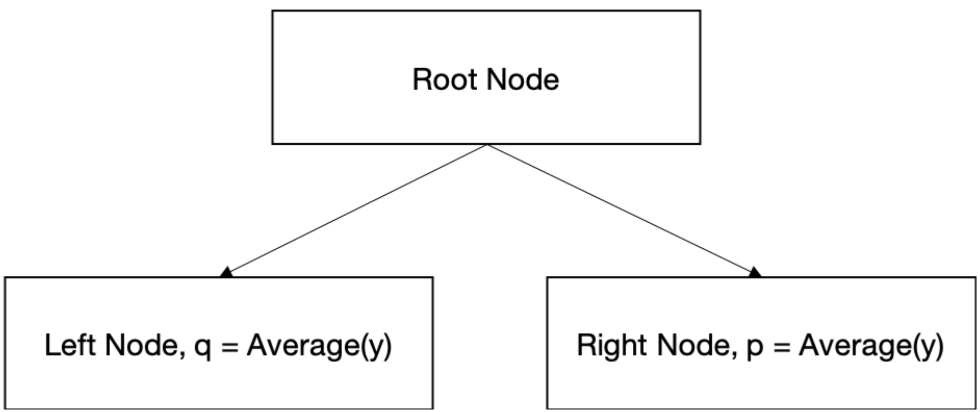
Decision Stump

Decision Stump is a 1-level Decision Tree.

Let's assume we have a **binary classification** dataset.

We pick $(x_{(j)}, v_s)$ to split the dataset into two partitions. In other words, we have $\mathcal{D}_L = \{(\mathbf{x}, y) | x_{(j)} \leq v_s\}$ (left node) and $\mathcal{D}_R = \{(\mathbf{x}, y) | x_{(j)} > v_s\}$ (right node).

For these partitions, define $q = \text{Average}(y|\mathcal{D}_L)$ and $p = \text{Average}(y|\mathcal{D}_R)$.



Log-likelihood of Decision Stump

Under this Decision Stump, we predict p if $x_{(j)} > v_s$ and q if $x_{(j)} \leq v_s$.

Let's assume $p > q$ i.e. the right node will be the positive prediction node.

We can draw a confusion table, and calculate the log-likelihood at each node.

$$\mathrm{LL}_L = \mathrm{FN} \times \log(q) + \mathrm{TN} \times \log(1 - q)$$

$$\mathrm{LL}_R = \mathrm{TP} \times \log(p) + \mathrm{FP} \times \log(1 - p)$$

	Left Node, $x_{(j)} \leq v_s$	**Right Node,** $x_{(j)} > v_s$
Negative Class ($y = 0$)	$\text{log-likelihood} = \mathrm{TN} \times \log(1 - q)$	$\mathrm{FP} \times \log(1 - p)$
Positive Class ($y = 1$)	$\mathrm{FN} \times \log(q)$	$\mathrm{TP} \times \log(p)$

Learning Decision Stump

Pick a splitting variable and value pair, $(x_{(j^*)}, v_{s^*})$, that gives the maximum log-likelihood.

$$(x_{(j^*)}, v_{s^*}) = \arg \max_{(x_{(j)}, v_s)} \mathrm{LL}_L + \mathrm{LL}_R$$

We search all possible variable and value combinations, and select the best pair.

This splitting criterion is called Information Gain.

$$H(y|\mathrm{Left}) = -q \log(q) - (1-q) \log(1-q)$$

$$H(y|\mathrm{Right}) = -p \log(p) - (1-p) \log(1-p)$$

$$\mathrm{InfoGain} = -\mathrm{size}_L H(y|\mathrm{Left}) - \mathrm{size}_R H(y|\mathrm{Right})$$

Note that maximum Information Gain is equivalent to maximum log-likelihood in this case.

Stump to Tree

To learn multi-level Decision Stump i.e. Decision Tree, we **recursively** iterate the same process on each partition.

```
def decision_tree(X, y, max_depth):

  n, m = X.shape
  if n < 3 or max_depth == 0:
    return np.mean(y)

  j_best, s_value = select_split_pair(X, y)

  left_idx = X[:,j_best] <= s_value
  right_idx = X[:,j_best] > s_value

  X_left, y_left = X[left_idx,:], y[left_idx]
  X_right, y_right = X[right_idx,:], y[right_idx]

  return {"split_var": j_best,
          "split_value": s_value,
          "left": decision_tree(X_left, y_left, max_depth-1),
          "right": decision_tree(X_right, y_right, max_depth-1)}
```

Other Splitting Criteria

Information Gain is just one way of selecting the splitting variable and value.

CART (Classification and Regression Tree) selects the splitting variable and value pair that mimizes Gini Impurity

$$\text{Gini}_L = 1 - q^2$$

$$\text{Gini}_R = 1 - p^2$$

$$\text{Gini Impurity} = \text{size}_L \text{Gini}_L + \text{size}_R \text{Gini}_R$$

Various other splitting criteria exist in the form of:

$$\min \text{size}_L h(q) + \text{size}_R h(p)$$

where $h(\cdot)$ is a function that measures the impurity of the target variable, y.

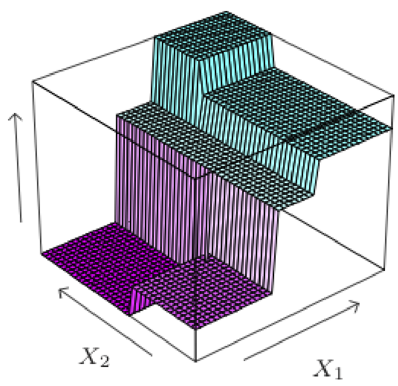
Regression Trees

Decision Tree can be applied to regression tasks.

To minimize Mean Squared Error, we find the best splitting pair that minimizes:

$$\text{size}_L \text{Var}(y|\text{Left}) + \text{size}_R \text{Var}(y|\text{Right})$$

CART uses this splitting criterion for regression tasks.



Chapter 9 of https://web.stanford.edu/~hastie/ElemStatLearn/

Handling Missing Values

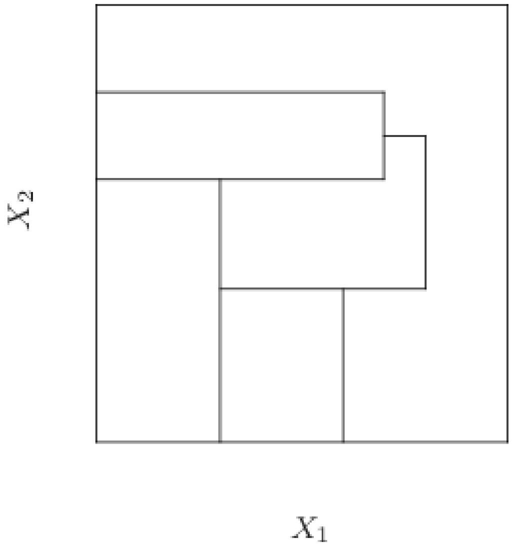
Missing Values can be tricky to deal with when using machine learning algorithms.

Many algorithms require either dropping samples with missing values or imputing missing values with some mechanisms.

Decision Trees have a few additional options to deal with missing values:

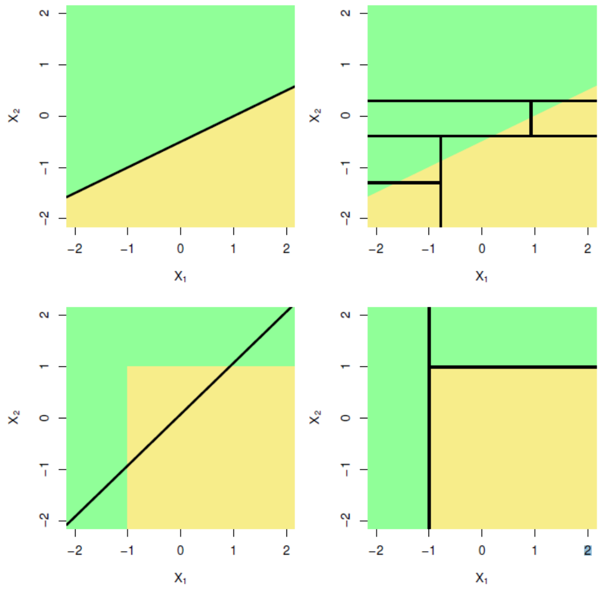
1. For a categorical variable, we can treat "missing" as an additional categorical value.
2. For a numeric variable, we can send samples with missing values to the child node (left or right) that gives better prediction i.e. surrogate splits for missing values

Limitations



Chapter 9 of https://web.stanford.edu/~hastie/ElemStatLearn/

Tree vs Linear Models



http://www.rnfc.org/courses/isl/Lesson%208/Videos/

Real-World Example

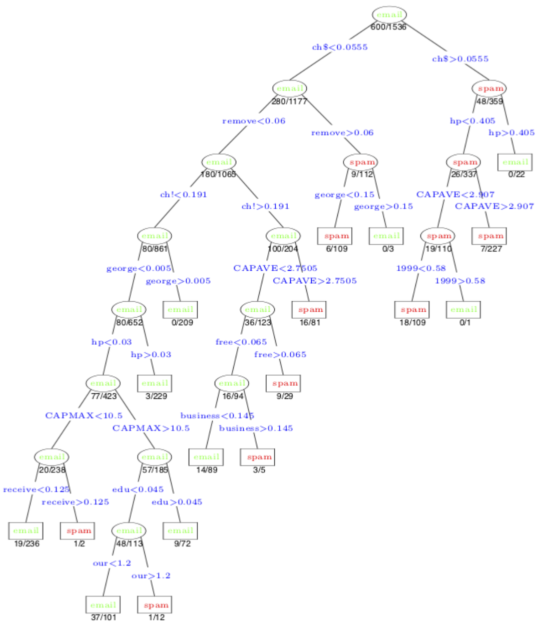


FIGURE 9.5. *The pruned tree for the* spam *example. The split variables are shown in blue on the branches, and the classification is shown in every node. The numbers under the terminal nodes indicate misclassification rates on the test data.*

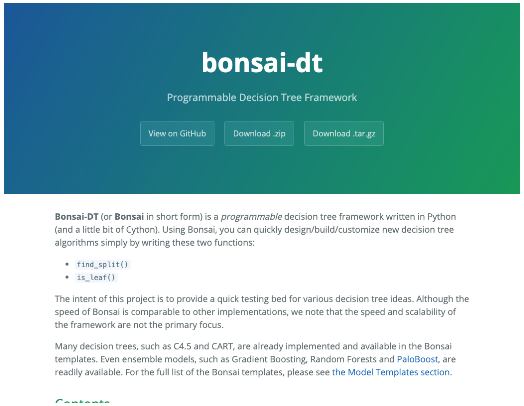
Chapter 9 of https://web.stanford.edu/~hastie/ElemStatLearn/

Bonsai-DT

Numerous decision tree algorithms out there.

- C4.5 uses Information Gain to split nodes
- CART uses Gini Impurity and Variance
- Hellinger Tree uses Hellinger Distance
- Alph-Tree uses Alpha-Divergence

You can make your decision tree by making a new splitting criterion (and that's it). This is my open-source project for easy-to-make custom decision trees. Check https://yubin-park.github.io/bonsai-dt/:



Questions?