# Decision Tree and Random Forest

Mobina Shahbandeh

University of Tehran ACM Summer School 2021

Playing golf according to weather, example from [http://www.saedsayad.com/zeror.htm](http://www.saedsayad.com/zeror.htm)

| Predictors | | | | Target |
|---|---|---|---|---|
| **Outlook** | **Temp.** | **Humidity** | **Windy** | **Play Golf** |
| Rainy | Hot | High | False | No |
| Rainy | Hot | High | True | No |
| Overcast | Hot | High | False | Yes |
| Sunny | Mild | High | False | Yes |
| Sunny | Cool | Normal | False | Yes |
| Sunny | Cool | Normal | True | No |
| Overcast | Cool | Normal | True | Yes |
| Rainy | Mild | High | False | No |
| Rainy | Cool | Normal | False | Yes |
| Sunny | Mild | Normal | False | Yes |
| Rainy | Mild | Normal | True | Yes |
| Overcast | Mild | High | True | Yes |
| Overcast | Hot | Normal | False | Yes |
| Sunny | Mild | High | True | No |

Decide whether one plays golf or not according to weather(yes or no).

Decision attributes are overlook, temp, humidity, windy and the frequency table is

| ★ | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Outlook | Sunny | 3 | 2 |
| | Overcast | 4 | 0 |
| | Rainy | 2 | 3 |

| | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Temp. | Hot | 2 | 2 |
| | Mild | 4 | 2 |
| | Cool | 3 | 1 |

| | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Humidity | High | 3 | 4 |
| | Normal | 6 | 1 |

| | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Windy | False | 6 | 2 |
| | True | 3 | 3 |

# Entropy

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

| Play Golf | |
|---|---|
| Yes | No |
| 9 | 5 |

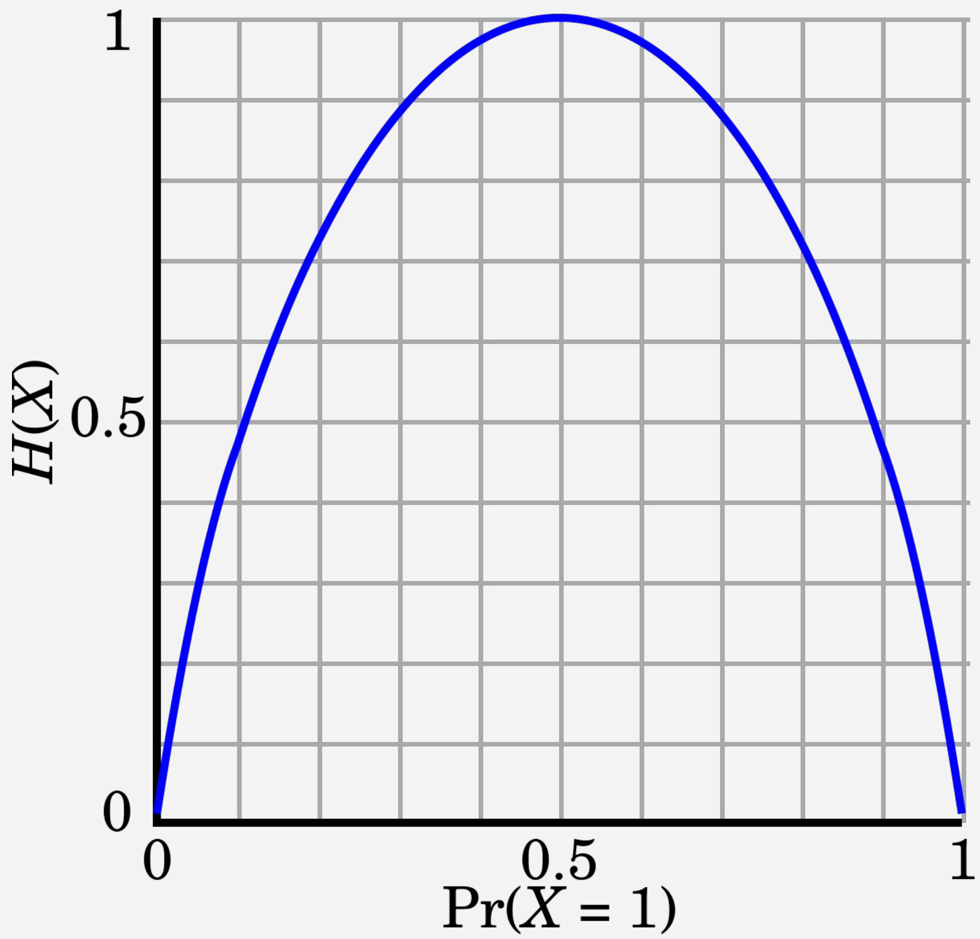
Entropy(PlayGolf) = Entropy (5,9)

$= \text{Entropy}(0.36, 0.64)$

$= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64)$

$= 0.94$

Entropy curve for binary class, q=1-p



$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

| | | Play Golf | | |
|---|---|---|---|---|
| | | Yes | No | |
| Outlook | Sunny | 3 | 2 | 5 |
| | Overcast | 4 | 0 | 4 |
| | Rainy | 2 | 3 | 5 |
| | | | | 14 |

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

# Information gain

$$\textbf{Gain} = \textbf{Entropy(T)} - \textbf{Entropy(T,X)}$$

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

**Step 1:** Calculate entropy of the target.

$$\begin{aligned}
\text{Entropy(PlayGolf)} &= \text{Entropy }(5,9) \\
&= \text{Entropy }(0.36,\ 0.64) \\
&= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\
&= 0.94
\end{aligned}$$

**Step 2:** The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

| | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Outlook | Sunny | 3 | 2 |
| | Overcast | 4 | 0 |
| | Rainy | 2 | 3 |
| Gain = 0.247 | | | |

| | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Temp. | Hot | 2 | 2 |
| | Mild | 4 | 2 |
| | Cool | 3 | 1 |
| Gain = 0.029 | | | |

| | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Humidity | High | 3 | 4 |
| | Normal | 6 | 1 |
| Gain = 0.152 | | | |

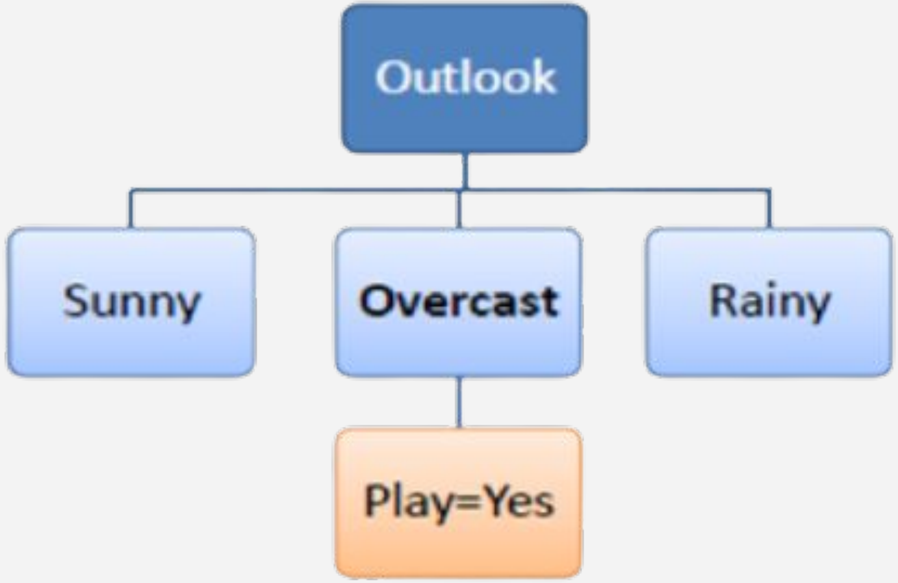
| | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Windy | False | 6 | 2 |
| | True | 3 | 3 |
| Gain = 0.048 | | | |

**Step 3:** Choose attribute with the largest information gain as the decision node.

| | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Outlook | Sunny | 3 | 2 |
| | Overcast | 4 | 0 |
| | Rainy | 2 | 3 |
| Gain = 0.247 | | | |

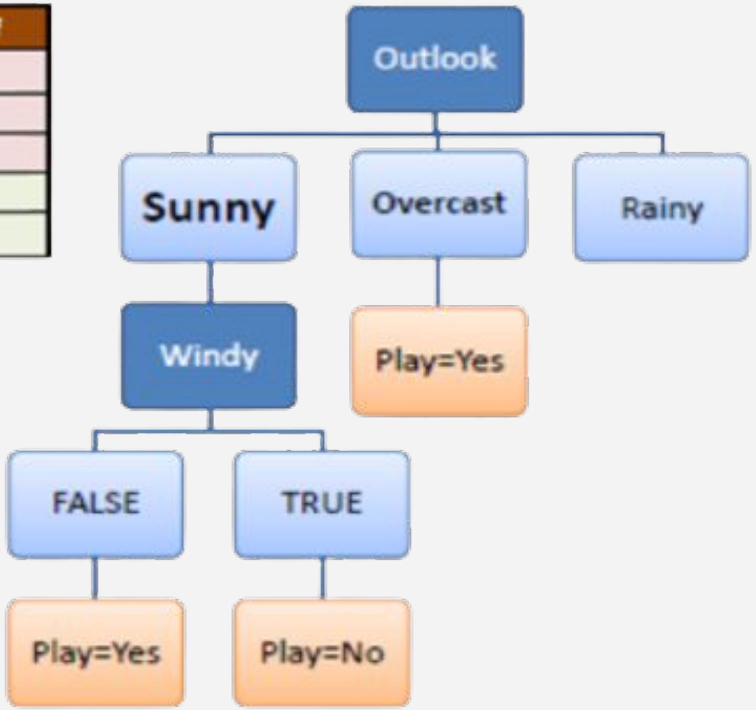
Step 4a: A branch with entropy of 0 is a leaf node.

| Temp | Humidity | Windy | Play Golf |
|---|---|---|---|
| Hot | High | FALSE | Yes |
| Cool | Normal | TRUE | Yes |
| Mild | High | TRUE | Yes |
| Hot | Normal | FALSE | Yes |
| Hot | High | FALSE | Yes |



**Step 4b:** A branch with entropy more than 0 needs further splitting.

| Temp | Humidity | Windy | Play Golf |
|---|---|---|---|
| Mild | High | FALSE | Yes |
| Cool | Normal | FALSE | Yes |
| Mild | Normal | FALSE | Yes |
| Cool | Normal | TRUE | No |
| Mild | High | TRUE | No |



**Step 5:** The algorithm is run recursively on the non-leaf branches, until all data is classified.

# Decision tree to decision rules

A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

$R_1$: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

$R_2$: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

$R_3$: IF (Outlook=Overcast) THEN Play=Yes

$R_4$: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

$R_5$: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes

# Random forest

In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, i.e. have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance of the final model.

# Tree bagging

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set X = x1, ..., xn with responses Y = y1, ..., yn, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, ..., B: 1.Sample, with replacement, B training examples from X, Y; call these Xb, Yb.

Train a decision or regression tree fb on Xb, Yb.

# Averaging

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

or by taking the majority vote in the case of decision trees.

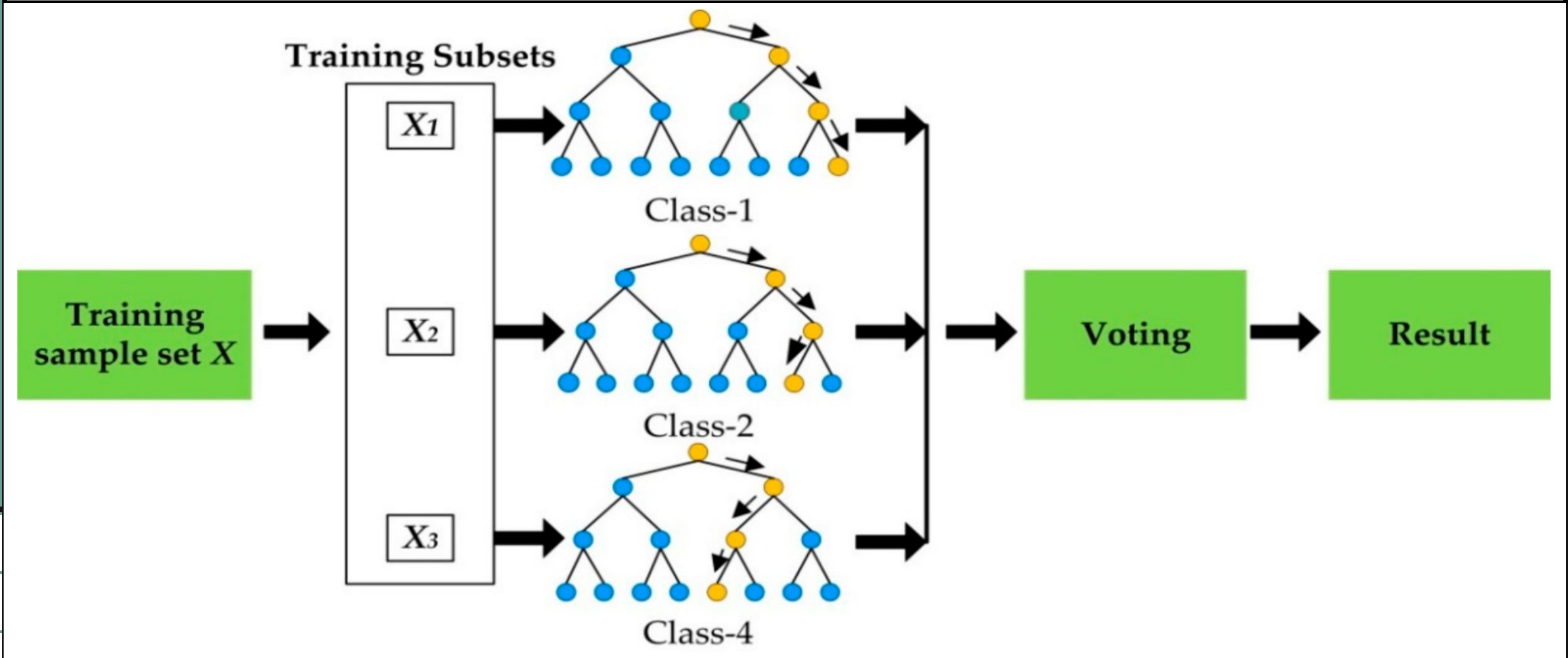
This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal number of trees B can be found using cross-validation, or by observing the out-of-bag error: the mean prediction error on each training sample $x_i$, using only the trees that did not have $x_i$ in their bootstrap sample. The training and test error tend to level off after some number of trees have been fit.

# From bagging to random forests

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated.

Typically, for a classification problem with p features, $\sqrt{p}$ (rounded down) features are used in each split. For regression problems the inventors recommend p/3 (rounded down) with a minimum node size of 5 as the default.


Training Subsets
Training sample set X
X1
X2
X3
Class-1
Class-2
Class-4
Voting
Result