CSE/ECE-471: Statistical Methods in AI **Spring 2020**

Decision Tree

Prepared by: Sai Kiran Velankayala(2019900079), Pranshi Yadav(2019202004), Sasanka Chekuri(2019900015)

In this note, we discuss introduction to decision trees and identifying trees in classification problems.

1 Background and Introduction

So, far we have discussed about classification techniques in Machine Learning like K-nearest neighbours. Lets discuss an example where nearest neighbours technique would face challenges and we would need a better approach for classification.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making.

Features:

1. It is one of the most intuitive classifiers
2. It is easy to understand and construct as well
3. Surprisingly, it also works very well

There are various examples where decision tree can be used as a model for classification:

1. Banks (eg. whether to provide loan or not)
2. Credit approval
3. Fruits classification
4. Identify whether a customer is buyer or not
5. Classify whether a mushroom is edible or not
6. Used for better movie recommendation, where a decision tree of all the watch-list is made based on the movies the user liked, and then a prediction is made as to what movie a user can watch next

2 The Vampire Analogy

Suppose, there's a little database of samples of individuals who have been determined to be vampires and not vampires as follows.

is Vampire ?	Casts Shadow ?	Eats Garlic ?	Skin Complexion	Accent
No	?	Yes	Pale	None
No	Yes	Yes	Rubby	None
Yes	?	No	Rubby	None
Yes	No	No	Average	Heavy
Yes	?	No	Average	Odd
No	Yes	No	Pale	Heavy
No	Yes	No	Average	Heavy
No	?	Yes	Rubby	Odd

**This data is for classroom illustration.In real world, good data sets doesn't occur.

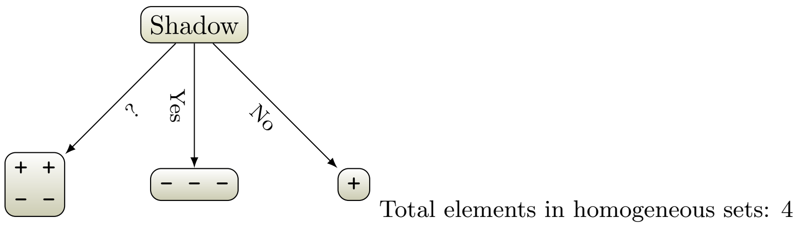
Given Table shows the data for classification of Vampires, based on the factors like Shadow, Garlic eating habits, Skin Complexion and Accent.It would be difficult to use nearest neighbors technique with data like this, due to the following problems.

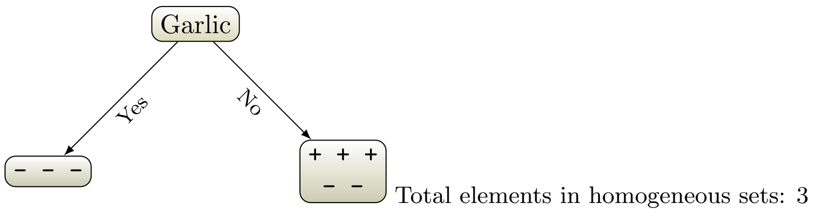
1. This is not numerical data.
2. It's not clear that all of these characteristics actually matter. So some characteristics may not correspond to the output of the classification.
3. Some characteristics do matter, but they only matter part of the time.
4. And finally, there's the matter of cost. Some of these tests may be more expensive to perform than others.

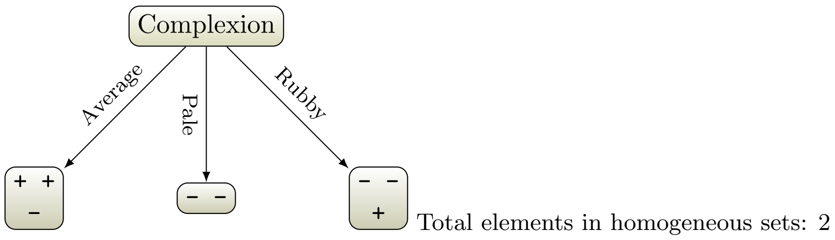
In such cases, we opt for a different model.

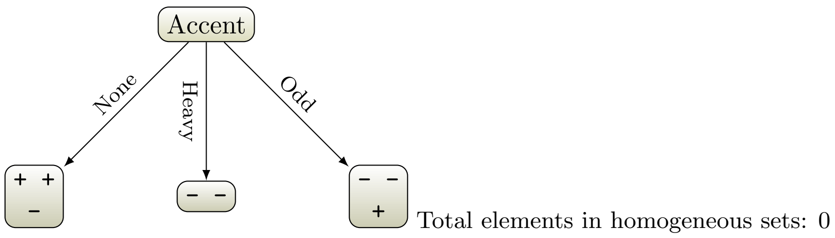
The first step and most important step is to decide which attribute is most important and that it should be the root node of the tree, and subsequently, we have to decide the attributes which are important for the classification in the order of importance.
For a particular node, there are finite choices to move ahead in the tree. This is a Boolean classification, so at the end of the decision tree, we would have two possible results (either they are a vampire or not). So, each example input will classify as true(a positive example) or false (a negative example).

Here, '+' refers to positive, which means that a person **is a vampire**, and '-' refers to negative, which means the person **is not a vampire**.



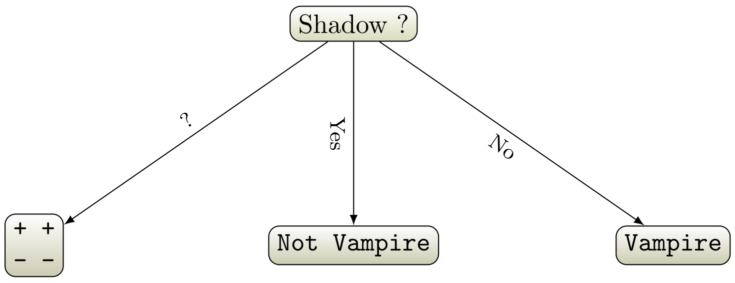






Ideally, we'd like a test that will put all the vampires in one group and all the ordinary people in another. Now, using greedy approach, lets pick the test with maximum homogeneous sets as root node.

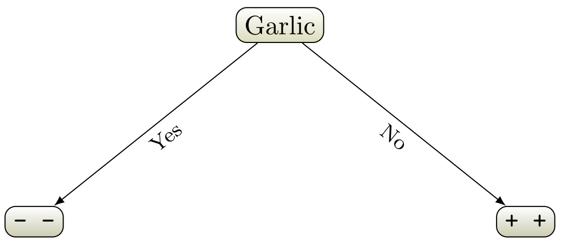
So, we get ..



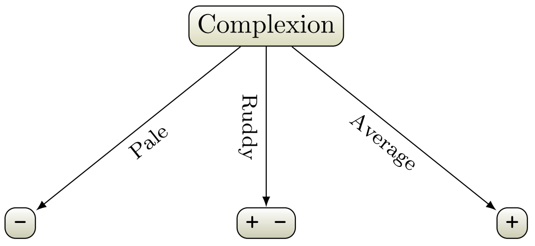
For the shadow attribute "yes" and "no", we can decide if a person is a vampire or not, but in case of "?" we don't know, so, we need to decide which attribute divides data well when shadow = '?'. For that, we need to analyze other attributes while shadow is unknown. (that is, consider only those tuples where shadow is labeled as '?')

Rebuilding the data table with rows when Shadow is unknown; we get ..

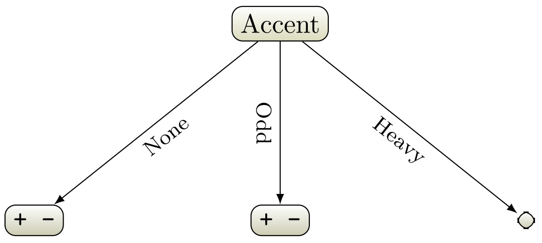
is Vampire ?	Casts Shadow ?	Eats Garlic ?	Skin Complexion	Accent
No	?	Yes	Pale	None
Yes	?	No	Rubby	None
Yes	?	No	Average	Odd
No	?	Yes	Rubby	Odd



Total elements in homogeneous sets: 4



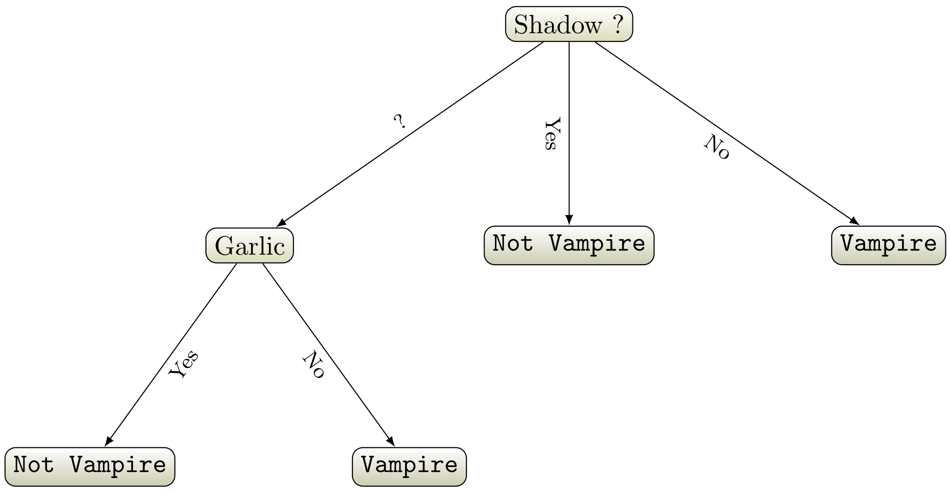
Total elements in homogeneous sets: 2



Total elements in homogeneous sets: 0

Here we find that "Garlic?" attribute divides maximum elements, in fact, all elements in homogeneous sets.

So, our decision tree now looks like this,



Problems while working with pure set:
This tree looks simpler and we observe that the greedy approach is helping us to get better results. But, is that the right way to do so?
No, because if the data-set is large, and there will be noise in the data, we need not end up with attributes dividing into the homogeneous set, we may find for all attributes elements in the homogeneous set are 0.
There is also a possibility that we find one good separator but other attributes gets messed up and give very low accuracy. While other permutations gives better accuracy. Hence, we need sets which are less distributed.

In such cases, how do we proceed to identify Trees ? Let's revise about the concept called **Entropy** before discussing the algorithmic approach.

3 Impurity functions

3.1 Entropy and Information gain

Entropy, as it relates to machine learning, is a measure of randomness in the information being processed.
The **higher** the entropy, the harder it is to draw any conclusion from the information.

Entropy can be defined as: $E = -\sum_{i=1}^{N} p_i log_2(p_i)$

where, p_i is the probability of each set in class
Let's take an example of a two-class problem : $\boxed{+ + - -}$

For this example, we can calculate entropy as:
$E_2 = -p_1 log_2(p_1) - p_2 log_2(p_2)$
here, p_1 represents positive class and p_2 represents negative class
So, the probability of getting a positive answer is half ($p_1 = \frac{2}{4} = \frac{1}{2}$)
Similarly for negative class, $p_2 = \frac{2}{4} = \frac{1}{2}$
Substituting these values in the entropy equation , we get,
$E_2 = -\frac{1}{2} log_2(\frac{1}{2}) - \frac{1}{2} log_2(\frac{1}{2})$
$E_2 = -\frac{1}{2}(-1) - \frac{1}{2}(-1) = 1$

This is the max(worst) entropy one can get. It means that there is 50% chance of getting each class. Entropy maximizes if data is distributed uniformly among classes.

Let's take another example for two class problem: $\boxed{- - -}$

$E_2 = -\frac{0}{3}log_2(\frac{0}{3}) - \frac{3}{3}log_2(\frac{3}{3})$
$E_2 = -(0)log_2(0) - (1)(0) = 0$
Here, we get can clearly predict the solution.

Note - [High E - Uniform Distribution - Unpredictability increases]

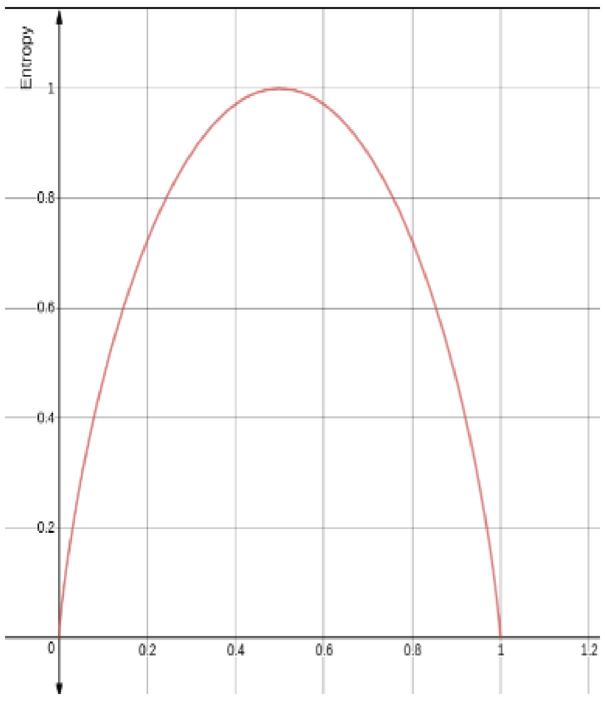


figure- Entropy graph

The information gain is based on the decease in entropy after a data-set is split on an attribute. Constructing a decision tree is all about finding the attribute that returns the highest information gain, i.e., the most homogeneous branch.
Steps to calculate information gain:

1. Calculate the entropy of the target(label)
2. Calculate the entropy of each branch of a split attribute. Then, calculate the weighted entropy sum of that attribute. This is done for each attribute split.
3. Calculate Information gain for each split
   Information gain = Entropy(target) - Weighted entropy sum
   Information gain is calculated for each split.
4. Attribute with highest information gain is selected as next node for split.
5. A branch with Entropy 0 is a leaf node and those with Entropy greater than 0 needs further splitting.
6. Repeat the same process for all non-leaf branches

(Refer section 5 for example)

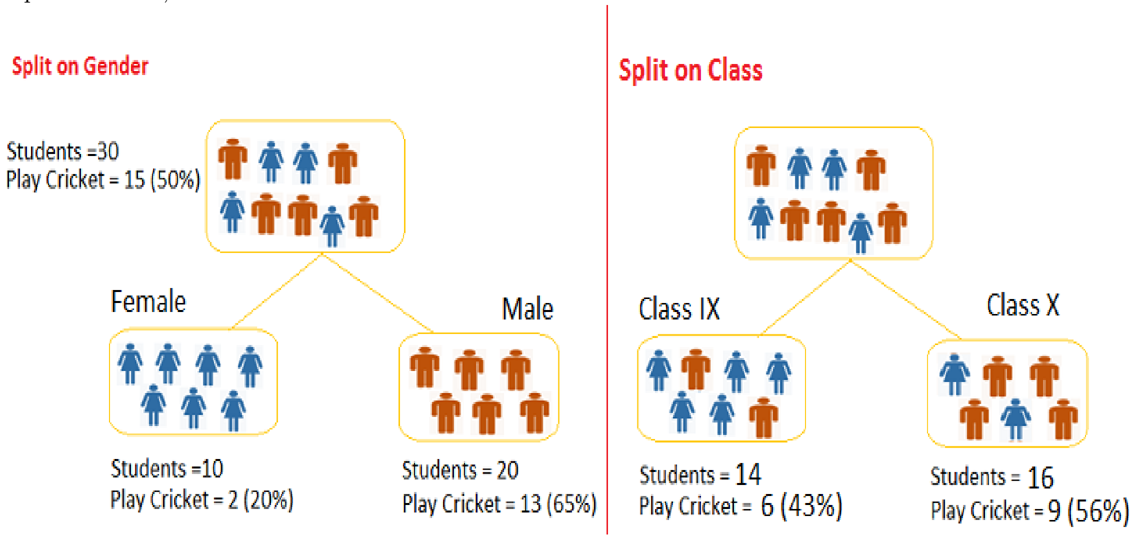
3.2 Gini Index

Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. Thats is, if we slect two items from a population at random then they must be of same class and the probability for this is 1 if population is pure.
An attribute with lower Gini index should be preferred.
Gini Index= $1 - \sum_{i=1}^{c} {p_i}^2$

Steps to calculate Gini Index:

1. Calculate the Gini for sub-nodes using the above formula
2. Calculate Gini for split using weighted score of each node of that split
3. Compare the Gini of all split and choose the one with least value

For example, we want to segregate whether the student plays cricket or not. Here, we split on the basis of two input variables, Gender and Class.



Now,
Split on Gender:

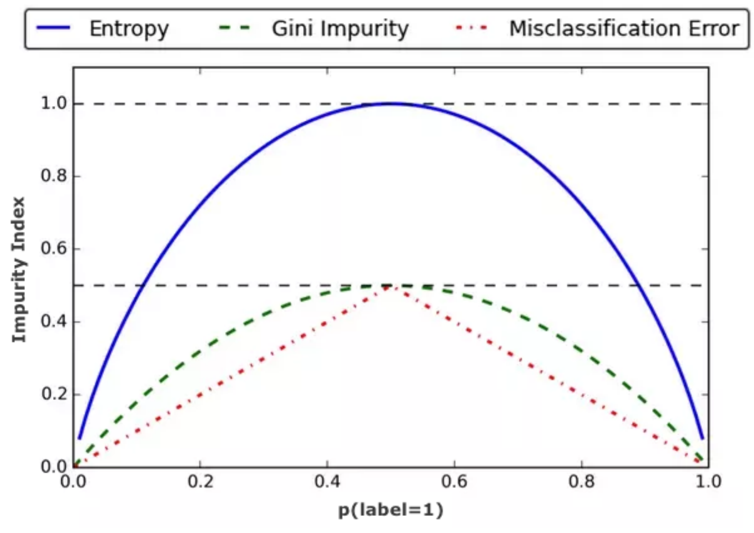
1. Gini for sub-node Female = 1- ((0.2)*(0.2) + (0.8)*(0.8)) = 1 - 0.68 = 0.32 (1- (success^2 + fail^2))
2. Gini for sub-node Male= 1 - ((0.65)*(0.65) + (0.35)*(0.35)) = 1 - 0.55 = 0.45
3. Weighted Gini for split Gender = $(\frac{10}{30})$*0.32 + $(\frac{20}{30})$*0.45 = 0.41

Split on Class:

1. Gini for sub-node class IX = 1- ((0.43)*(0.43) + (0.57)*(0.57)) = 1 - 0.51 = 0.49
2. Gini for sub-node class X= 1 - ((0.56)*(0.56) + (0.44)*(0.44)) = 1 - 0.51 = 0.49
3. Weighted Gini for split class = $(\frac{14}{30})$*0.49 + $(\frac{16}{30})$*0.49 = 0.49

From above, we can see that gender has less Gini value. So, we will split on the basis of gender here.

4 Analysis of impurity functions



Gini impurity measures how heterogeneous or mixed some value is over a set. In decision trees, we have to find a criteria that make a set into more homogeneous subsets. Information gain is often used to describe this difference that's being maximized though that term goes with entropy. But it's really entropy that is the alternative to Gini impurity.The higher the score the higher the impurity and therefore the higher the information gain will be when a split occurs.

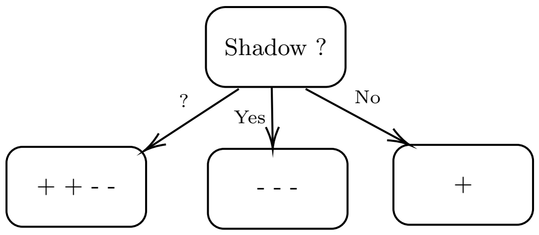
5 Solution to 'Vampire or not' problem using Information Gain

Now, we'll see how information gain can be used to create decision tree.
We need to find attribute that returns the highest information gain or minimum disorder(Weighted Entropy Sum). Lets calculate them for each of the attribute : Shadow, Garlic, Complexion, Accent, using the following formula. Let P and N represent the number of '+' and '-'.

$$Entropy\ (E_i)\ = -\ \frac{P}{(P+N)}\ log\ \left(\frac{P}{(P+N)}\right) -\ \frac{N}{(P+N)}\ log\ \left(\frac{N}{(P+N)}\right)$$

$$Weighted\ Entropy\ Sum\ = \sum\ W_i E_i \qquad ;\ W_i\ is\ the\ weight$$

$$Information\ Gain(Attribute)\ =\ Entropy\ of\ attribute\ -\ Weighted\ average\ of\ Entropy\ of\ each\ child\ set$$



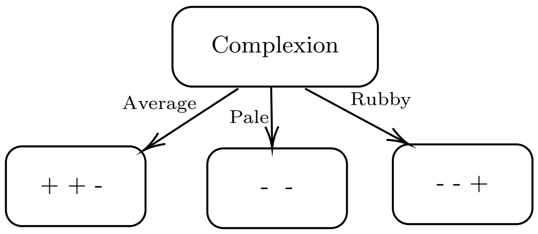
P=2, N=2 P=0, N=3 P=1, N=0

$E1 = \frac{-1}{2} log\ \frac{1}{2}\ - \frac{1}{2} log\ \frac{1}{2}$ $E2 = 0 log\ \frac{0}{3}\ - 1 log\ \frac{3}{3}$ $E3 = 0 log\ \frac{0}{1}\ - 1 log\ \frac{1}{1}$

$\implies E1 = 1$ $\implies E2 = 0$ $\implies E3 = 0$

$$Weighted\ Entropy\ Sum\ =\ \frac{4}{8}\ *\ E1\ + \frac{3}{8}\ *\ E2\ + \frac{1}{8}\ *\ E3\ =\ \frac{1}{2}$$

$$Information\ Gain\ =\ 0.9544\ -\ 0.5\ \ =\ 0.4544$$



P=2, N=1 P=0, N=2 P=1, N=2

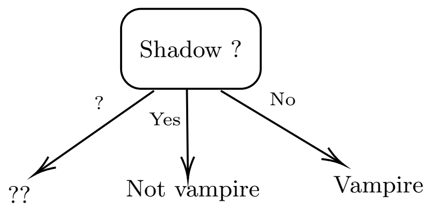
$E1 = \frac{-2}{3} log\ \frac{2}{3} - \frac{1}{3} log\ \frac{1}{3}$ $E2 = 0 log\ \frac{0}{2}\ - 1 log\ \frac{2}{2}$ $E3 = -\frac{1}{3} log\ \frac{1}{3}\ - \frac{2}{3} log\ \frac{2}{3}$

$\implies E1 = 0.918$ $\implies E2 = 0$ $\implies E3 = 0.918$

$$Weighted\ Entropy\ Sum\ =\ \frac{3}{8}\ *\ E1\ + \frac{2}{8}\ *\ E2\ + \frac{3}{8}\ *\ E3\ =\ 0.688$$

$$Information\ Gain\ =\ 0.9544\ -\ 0.688\ \ =\ 0.2656$$

Similarly, we get Weighted Entropy Sum of Garlic and Accent as 0.6 and 0.95 respectively, and the corresponding Information Gain as 0.3544 and 0.004. Now lets pick the attribute Shadow as the root node since , its has the maximum Information Gain.So choosing shadow as root attribute, we get ..



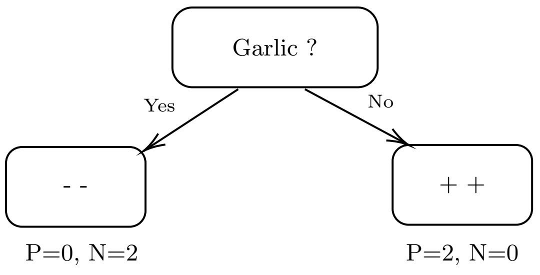
Now, We need to decide another attribute for Shadow = '?'. Again Rebuilding the data table with rows when Shadow is unknown; we get ..

is Vampire ?	Casts Shadow ?	Eats Garlic ?
No	?	Yes
Yes	?	No
Yes	?	No
No	?	Yes

Using the same approach discussed above, we get ..

$E1 = -\frac{0}{2} log\ \frac{0}{2} - \frac{2}{2} log\ \frac{2}{2}$

$\implies E1 = 0$



$E2 = -\frac{2}{2} log\ \frac{2}{2}\ - \frac{0}{2} log\ \frac{0}{2}$

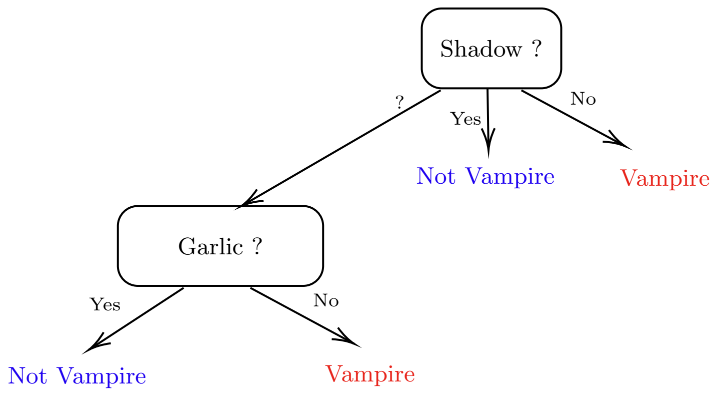
$\implies E2 = 0$

$$Entropy(Garlic\ when\ Shadow\ =\ "?")\ = 1$$

$$Weighted\ Average\ =\ \frac{2}{4}\ *\ E1\ +\ \frac{2}{4}\ *\ E2\ = 0$$

$$Information\ Gain\ =\ 1 - 0\ =\ 1$$

We can see that the Information Gain is maximum (=1) for Garlic attribute when shadow is unknown(=?). So, we pick Garlic as next attribute and our tree will look like this..



We need to note that this tree is exactly the same as the previous approach(greedy), because luckily at each step we were able to find some attributes dividing into a homogeneous set, but the approach with Information Gain is more robust, which can be applied to make a decision tree from a large data-set.

6 Decision Tree Regression

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output.

For regression tree generation, we calculate the coefficient of deviation (Standard deviation/Mean) for each split.

Steps:

1. Calculate Standard deviation of the label(target)
   $S = \sqrt{\frac{\sum(x-\overline{x})^2}{n}}$
2. Standard deviation of each attribute is calculated(for two values, attribute and label)
   $S(T,X) = \sum_{c \in X}(P(c))S(c)$
3. For each attribute, calculate Standard Deviation Reduction as
   SDR(T,X) = S(T) - S(T,X)
4. The attribute with the largest standard deviation reduction is selected for split.
5. The dataset is divided on the values of the selected attribute. This process is repeated for non-leaf branches.

We can stop the process when the Coefficient of deviation is below certain threshold. Finally, if the number of instances is more than one at a leaf node, calculate the mean as the final predicted value.

Note: each side of split will be a different test.

Problem with the model: In some cases, even if one point is changed even a little, the tree changes significantly.

7 Random Forest overview

Random Forest grows many classification trees. If we want to classify a new object from an input vector, put the vector into each tree of the forest. Each tree gives a classification, and we take tree votes from each tree and get the most vote and classify the object in that class.

Grow k different trees:

1. Pick a random subset of train data (with replacement)
2. Pick d random attributes
3. Compute gain based on subset
4. Repeat k times

Random forests are among the most widely used machine learning algorithm, probably due to their relatively good performance and ease of use (not much tuning required to get good results).

The random forest algorithm can be understood as bagging with decision trees,but instead of growing the decision trees by basing the splitting criterion on the complete feature set, we use random feature subsets.

Why random forest may work better ? The random forest good performance may be explained by the additional randomization that further diversifies the individual trees. Law of Large Numbers they do not over fit.

Breiman's random forest paper https://www.stat.berkeley.edu/ breiman/randomforest2001.pdf

The upper bound of the generalization error, where randomization of the feature sub spaces may decrease the "strength" of the individual trees, but at the same time, it reduces the correlation of the trees.

To summarize random forest, we fit decision trees on different samples and for each decision tree, we select a random subset of features at each node to decide upon the optimal split. while the size of the feature subset to consider at each node is a hyper parameter that we can tune, a “rule-of-thumb” suggestion is to use NumFeatures = log2m+ 1.

References

[1] Artificial Intelligence by Patrick Henry Winston, Link: https://courses.csail.mit.edu/6.034f/ai3/rest.pdf