Introduction to Decision Tree

A decision tree is one of the popular and powerful machine learning algorithms that I have learned. It is a non-parametric supervised learning method that can be used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. For a classification model, the target values are discrete in nature, whereas, for a regression model, the target values are represented by continuous values.

Decision Tree follow Greedy Algo

Programatically speaking, Decision trees are nothing but a giant structure of nested if-else condition

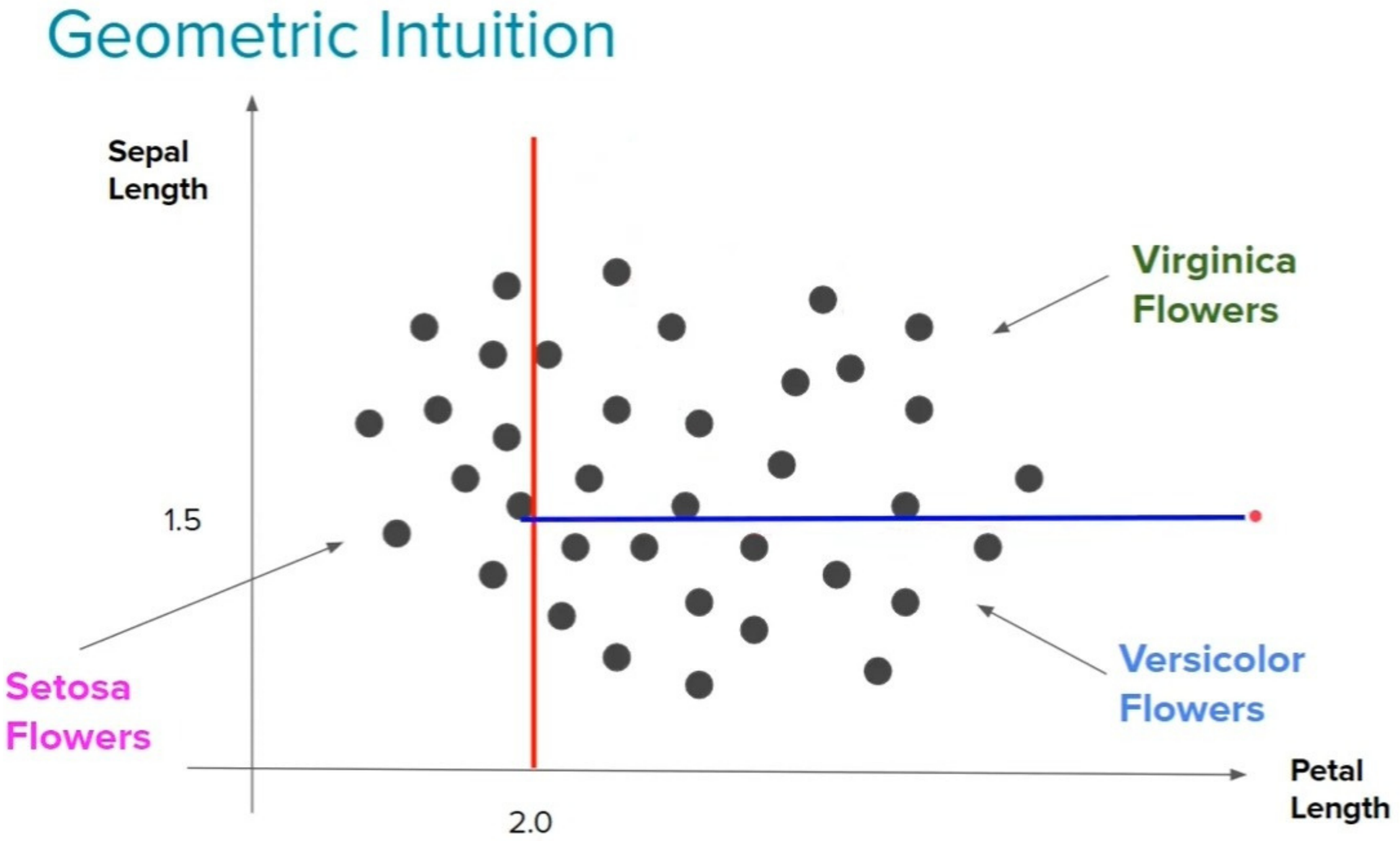
Mathematically speaking, Decision trees use hyperplanes which run parallel to any one of the axes to cut your coordinate system into hyper cuboids

Introduction to Decision Tree

A decision tree is a tree where each internal node specifies a test on an attribute of the data in question and each edge between a parent and a child represents a decision or outcome based on that test. Leaf nodes, sometimes called terminal nodes, contain classes or labels of the data. Tests are encoded as numerical or categorical rules such as “X > 5” or “X is ‘female’”. A decision tree can also be thought of as an apparatus that accepts as a set of input values and follows decision rules to get to a leaf of the tree, which corresponds to an output value.

The most basic process of training a decision tree on a dataset involves these three elements: the selection of attribute splits in the tree, the decision of when to stop splitting a node and mark it terminal, and the assignment of a label to each terminal node. Some algorithms add a fourth element, called pruning

Decision Tree (Classifiation)

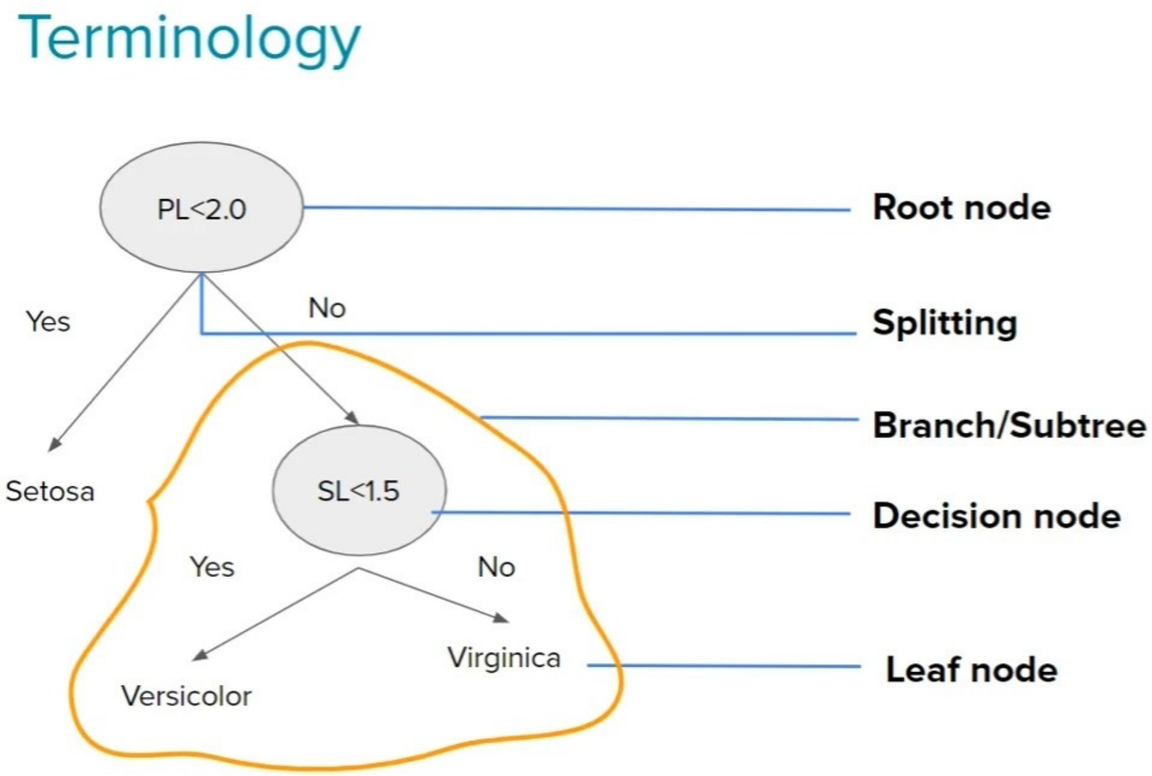


Introduction to Decision Tree

Pseudo code

- Begin with your training dataset, which should have some feature variables and classification or regression output.
- Determine the “best feature” in the dataset to split the data on; more on how we define “best feature” later
- Split the data into subsets that contain the correct values for this best feature. This splitting basically defines a node on the tree i.e each node is a splitting point based on a certain feature from our data.
- Recursively generate new tree nodes by using the subset of data created from step 3.

Introduction to Decision Tree



Root Node: It represents entire population or sample and this further gets divided into two or more homogeneous sets.

Splitting: It is a process of dividing a node into two or more sub-nodes.

Decision Node: When a sub-node splits into further sub-nodes, then it is called decision node.

Leaf / Terminal Node: Nodes do not split is called Leaf or Terminal node.

Pruning: When we remove sub-nodes of a decision node, this process is called pruning.

- Branch / Sub-Tree: A sub section of entire tree is called branch or sub-tree.

Parent and Child Node: A node, which is divided into sub-nodes is called parent node of sub-nodes where as sub-nodes are the child of parent node.

Introduction to Decision Tree

Some unanswered questions

How to decide which column should be considered as root node?

How to select subsequent decision nodes?

How to decide splitting criteria in case of numerical columns?

Introduction to Decision Tree

Advantages

Intuitive and easy to understand

Minimal data preparation is required

The cost of using the tree for inference is **logarithmic** in the number of data points used to train the tree

Disadvantages

Overfitting

Prone to errors for imbalanced datasets

How do Decision Trees work?

The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria are different for classification and regression trees.

Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that the purity of the node increases with respect to the target variable. The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

he algorithm selection is also based on the type of target variables. Let us look at some algorithms used in Decision Trees:

ID3 → (extension of D3)
C4.5 → (successor of ID3)
CART → (Classification And Regression Tree)
CHAID → (Chi-square automatic interaction detection Performs multi-level splits when computing classification trees)
MARS → (multivariate adaptive regression splines)

Steps in ID3 algorithm:

1. It begins with the original set S as the root node.
2. On each iteration of the algorithm, it iterates through the very unused attribute of the set S and calculates Entropy(H) and Information gain(IG) of this attribute.
3. It then selects the attribute which has the smallest Entropy or Largest Information gain.
4. The set S is then split by the selected attribute to produce a subset of the data.
5. The algorithm continues to recur on each subset, considering only attributes never selected before.

Attribute Selection Measures

1. Entropy,
2. Information gain,
3. Gini index,
4. Gain Ratio,
5. Reduction in Variance
6. Chi-Square

These criteria will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order i.e, the attribute with a high value(in case of information gain) is placed at the root.

While using Information Gain as a criterion, we assume attributes to be categorical, and for the Gini index, attributes are assumed to be continuous.

Impurity Funtion

An impurity function of a probability distribution P is a function ϕ: $P \rightarrow \mathbb{R}$ that satisfies
a few constraints:
1) ϕ attains its maximum if and only if P is a uniform distribution.
2) ϕ attains its minimum if and only if some $pj = 1$ for some $pj \in P$
3) ϕ is symmetric about the components of P
4) $\phi(P) \geq 0$ for all $\square$

Entropy

Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information.

entropy could be used to describe the amount of unpredictability in a random variable.

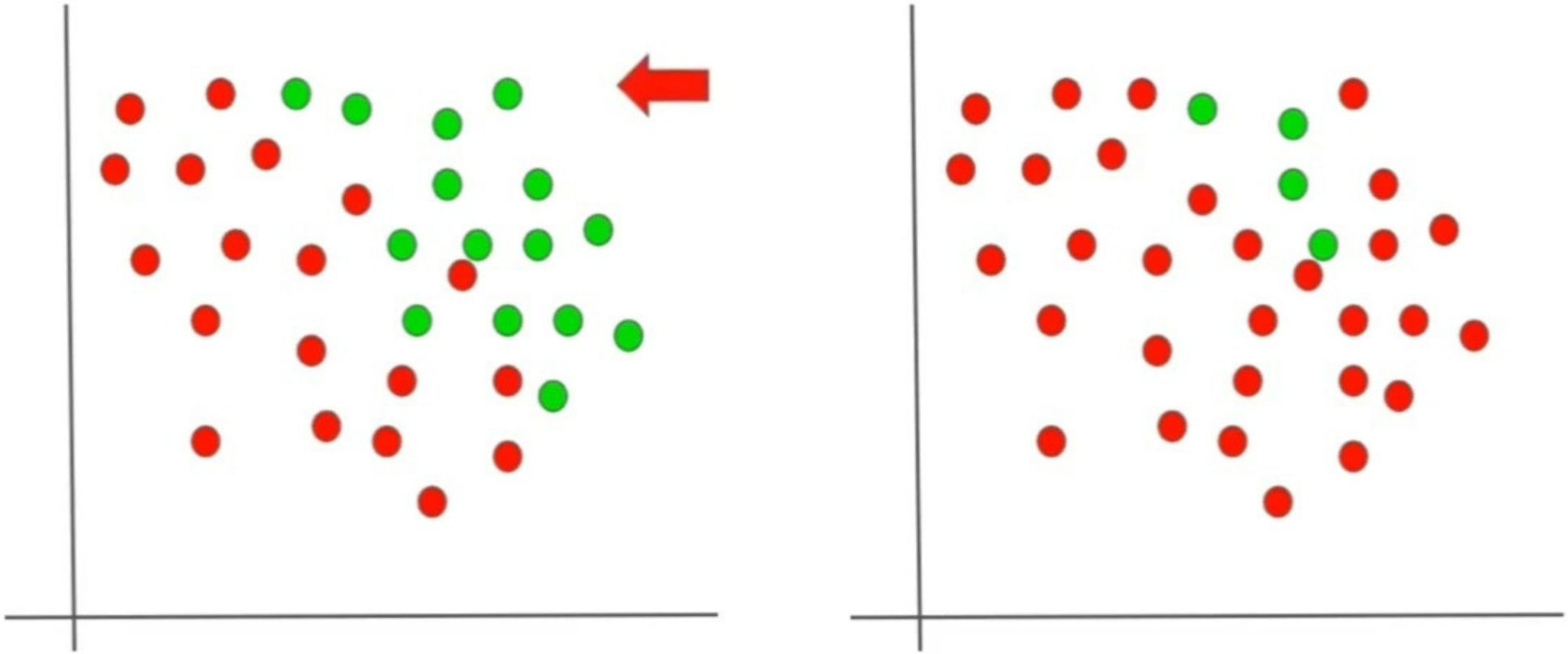
Entropy

Suppose you have a group of friends who decides which movie they can watch together on Sunday. There are 2 choices for movies, one is “Lucy” and the second is “Titanic” and now everyone has to tell their choice. After everyone gives their answer we see that “Lucy” gets 4 votes and “Titanic” gets 5 votes. Which movie do we watch now? Isn’t it hard to choose 1 movie now because the votes for both the movies are somewhat equal.

This is exactly what we call disorderness, there is an equal number of votes for both the movies, and we can’t really decide which movie we should watch. It would have been much easier if the votes for “Lucy” were 8 and for “Titanic” it was 2. Here we could easily say that the majority of votes are for “Lucy” hence everyone will be watching this movie

Entropy

Example 2 - Scatterplot



More knowledge less entropy

How to Calculate Entropy

The mathematical formula for entropy is:

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

Where ‘Pi’ is simply the frequentist probability of an element/class ‘i’ in our data.

For e.g if our data has only 2 class labels **Yes** and **No.**

$$E(D) = -p_{yes} \log_2(p_{yes}) - p_{no} \log_2(p_{no})$$

How to Calculate Entropy

Example 3 - Dataset

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
12000	18	No

Salary	Age	Purchase
34000	31	No
15000	25	No
69000	57	Yes
25000	21	No
32000	28	No

$H(d) = -P_y \log_2(P_y) - P_n \log_2(P_n)$

$H(d) = -2/5 \log_2(2/5) - 3/5 \log_2(3/5)$

$H(d) = 0.97$

$H(d) = -P_y \log_2(P_y) - P_n \log_2(P_n)$

$H(d) = -1/5 \log_2(1/5) - 4/5 \log_2(4$

$H(d) = 0.72$

Calculating entropy for a 3 class problem

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
30000	30	Maybe
12000	18	No

$H(d) = - P_y \log_2(P_y) - P_n \log_2(P_n) - P_m \log_2(P_m)$

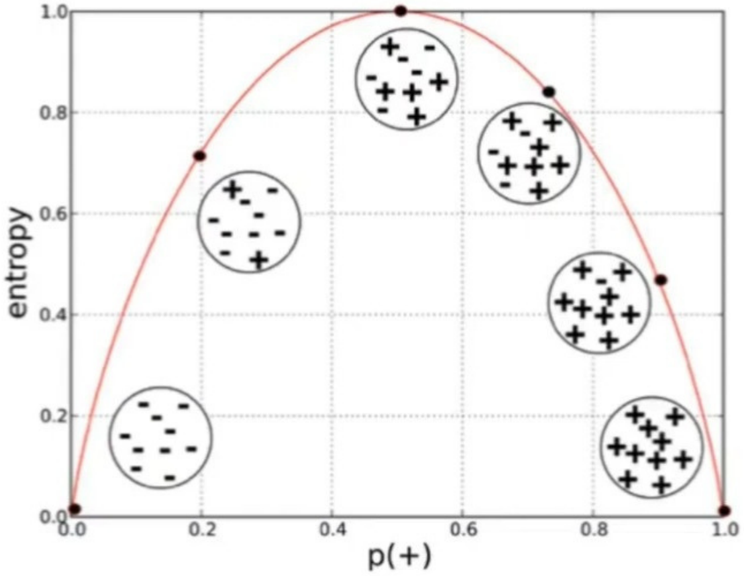
$H(d) = -2/8 \log_2(2/8) - 3/8 \log_2(3/8) - 3/8 \log_2$

$H(d) = 1.56$

Observation

- More the uncertainty more is entropy
- For a 2 class problem the min entropy is 0 and the max is 1
- For more than 2 classes the min entropy is 0 but the max can be greater than 1
- Both $\log_2$ or $\log_e$ can be used to calculate entropy

Entropy Vs Probability



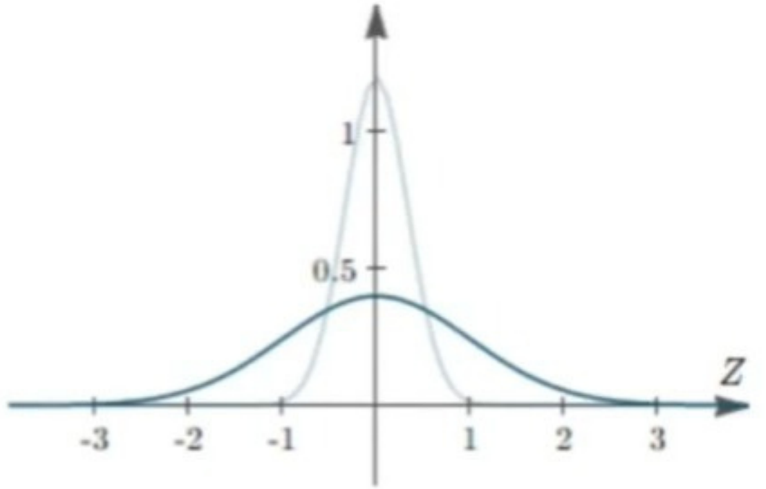
Entropy for Continuous Variables

Area	Built in	Price
1200	1999	3.5
1800	2011	5.6
1400	2000	7.3
...	...	...

Dataset 1

Area	Built in	Price
2200	1989	4.6
800	2018	6.5
1100	2005	12.8
...	...	...

Dataset 2



Distribution of Price

Quiz: Which of the above datasets have higher entropy?

Ans: Whichever is less peaked

Information Gain

Entropy gives measure of impurity in a node. In a decision tree building process, two important decisions are to be made — what is the best split(s) and which is the best variable to split a node.

Information Gain criteria helps in making these decisions

Information gain or IG is a statistical property that measures how well a given attribute separates the training examples according to their target classification. Constructing a decision tree is all about finding an attribute that returns the highest information gain and the smallest entropy.

$$Information\ Gain\ =\ Entropy(before) - \sum_{j=1}^{K} Entropy(j,\ after)$$

Entropy is used when determining how much information is encoded in a particular decision. This information gain is useful when, upon being presented with a set of attributes about your random variable, you need to decide on which attribute tells you the most info about the variable.

Gain Ratio

Gain ratio
Information gain is biased towards choosing attributes with a large number of values as root nodes. It means it prefers the attribute with a large number of distinct values.
C4.5, an improvement of ID3, uses Gain ratio which is a modification of Information gain that reduces its bias and is usually the best option. Gain ratio overcomes the problem with information gain by taking into account the number of branches that would result before making the split. It corrects information gain by taking the intrinsic information of a split into account.

Let us consider if we have a dataset that has users and their movie genre preferences based on variables like gender, group of age, rating, blah, blah. With the help of information gain, you split at ‘Gender’ (assuming it has the highest information gain) and now the variables ‘Group of Age’ and ‘Rating’ could be equally important and with the help of gain ratio, it will penalize a variable with more distinct values which will help us decide the split at the next level.

$$Gain\ Ratio = \frac{Information\ Gain}{SplitInfo} = \frac{Entropy\ (before) - \sum_{j=1}^{K} Entropy(j, after)}{\sum_{j=1}^{K} w_j \log_2 w_j}$$

Reduction in Variance

Reduction in variance is an algorithm used for continuous target variables (regression problems). This algorithm uses the standard formula of variance to choose the best split. The split with lower variance is selected as the criteria to split the population:

Above X-bar is the mean of the values, X is actual and n is the number of values.

Steps to calculate Variance:

1. Calculate variance for each node.
2. Calculate variance for each split as the weighted average of each node variance.

Chi-squared Values

The acronym CHAID stands for Chi-squared Automatic Interaction Detector. It is one of the oldest tree classification methods. It finds out the statistical significance between the differences between sub-nodes and parent node. We measure it by the sum of squares of standardized differences between observed and expected frequencies of the target variable.

It works with the categorical target variable “Success” or “Failure”. It can perform two or more splits. Higher the value of Chi-Square higher the statistical significance of differences between sub-node and Parent node.

It generates a tree called CHAID (Chi-square Automatic Interaction Detector).

Mathematically, Chi-squared is represented as:

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

Where:

χ^2 = Chi Square obtained
$\sum$ = the sum of
O = observed score
E = expected score

Information Gain

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Step 1:

Entropy of Parent

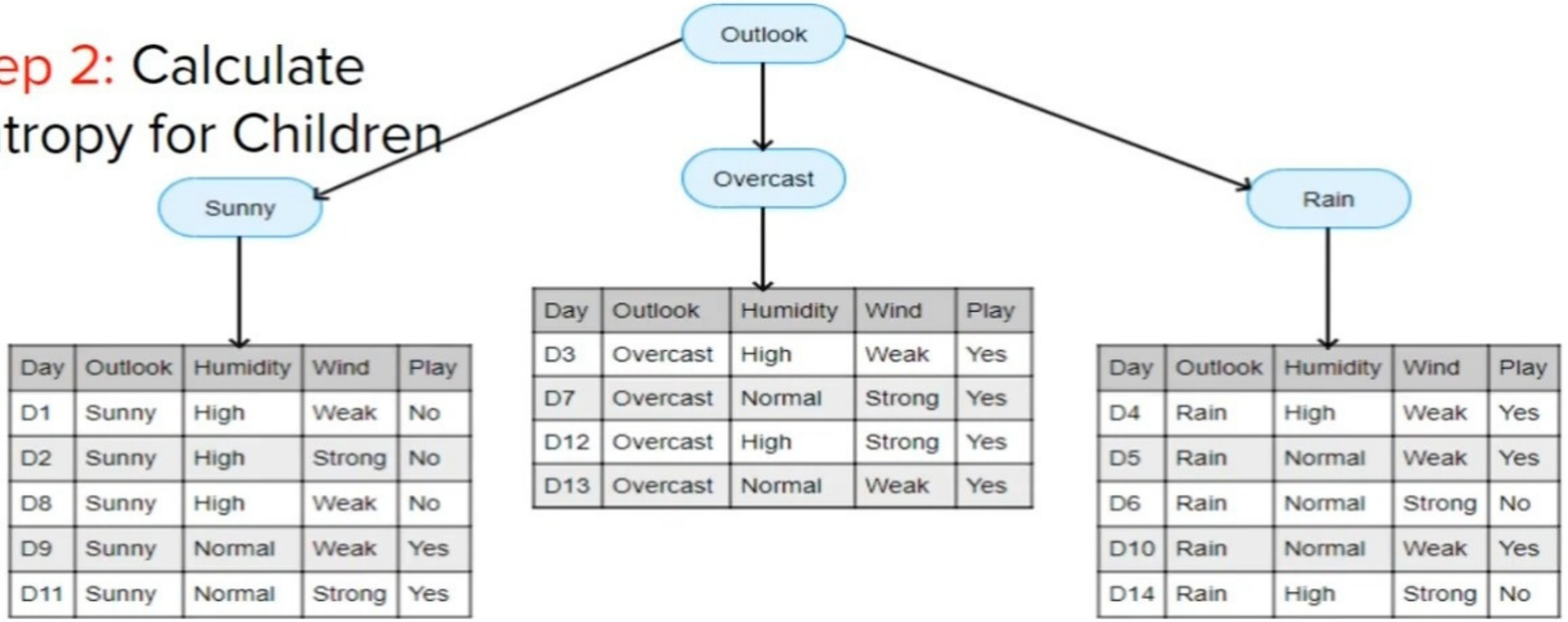
$E(P) = -p_y \log_2(p_y) - p_n \log_2(p_n)$

$= 9/14 \log_2(9/14) - 5/14/\log_2(5/14)$

$E(P) = \mathbf{0.94}$

Information Gain

Step 2: Calculate Entropy for Children



Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes

Day	Outlook	Humidity	Wind	Play
D3	Overcast	High	Weak	Yes
D7	Overcast	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes

Day	Outlook	Humidity	Wind	Play
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D10	Rain	Normal	Weak	Yes
D14	Rain	High	Strong	No

E(S) =
-2/5log(2/5) - 3/5log(3/5)

E(S)= 0.97

E(O) =
-5/5log(5/5) - 0/5log(0/5)

E(O)= 0

E(R) =
-3/5log(3/5) - 2/5log(2/5)

E(S)= 0.97

Information Gain

Step 3 : Calculate weighted Entropy of Children

Weighted Entropy = 5/14 * 0.97 + 4/14 * 0 + 5/14 * 0.97

W.E(Children) = **0.69**

P(Overcast) is a leaf node as it's entropy is 0

Information Gain

Step 4 : Calculate Information Gain

Information Gain = E(Parent) - {Weighted Average} * E(Children)

IG = **0.97 - 0.69 = 0.28**

So the information gain(or the decrease in entropy/impurity) when you split this data on the basis of **Outlook** condition/column is **0.28**

Information Gain

Step 5 : Calculate Information Gain for all the columns

Whichever column has the highest Information Gain(maximum decrease in entropy) the algorithm will select that column to split the data.

Step 6 : Find Information Gain recursively

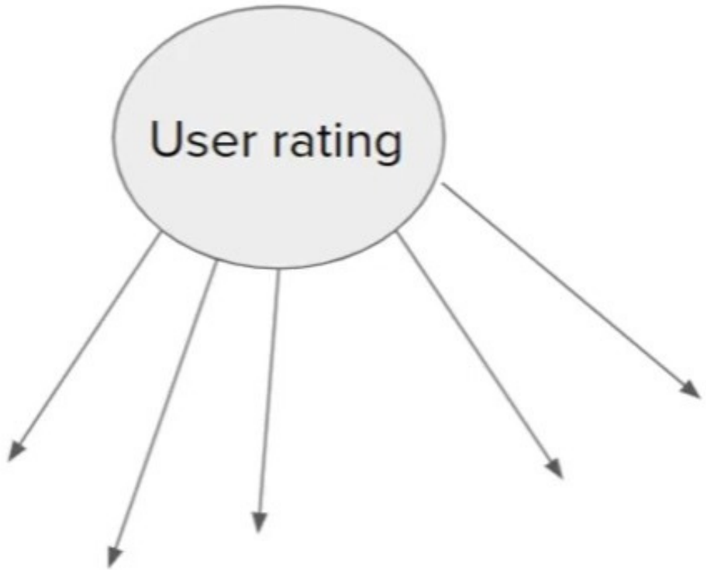
Decision tree then applies a recursive greedy search algorithm in top bottom fashion to find Information Gain at every level of the tree.

Once a leaf node is reached (Entropy = 0), no more splitting is done.

For Numerical Datasets

S No	User Rating	Downloaded
1	3.5	Yes
2	4.6	Yes
3	2.2	No
4	1.6	Yes
5	4.1	No
6	3.9	No
7	3.2	No
9	2.9	Yes
10	4.8	Yes
11	3.3	No
12	2.5	Yes
13	1.9	Yes

data['user_rating'].nunique()=n



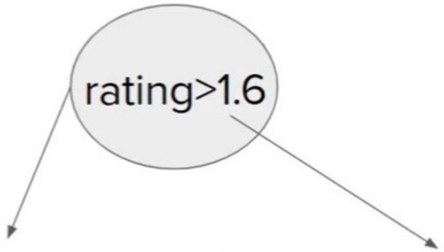
We will then have n subtrees

For Numerical Datasets

Step 1:
Sort the data on the basis of numerical column

S No	User Rating	Downloaded
1	1.6	Yes
2	1.9	Yes
3	2.2	No
4	2.5	Yes
5	2.9	Yes
6	3.2	No
7	3.3	No
9	3.5	Yes
10	3.9	No
11	4.1	No
12	4.6	Yes
13	4.8	Yes

Step 2:
Split the entire data on the basis of every value of user_rating



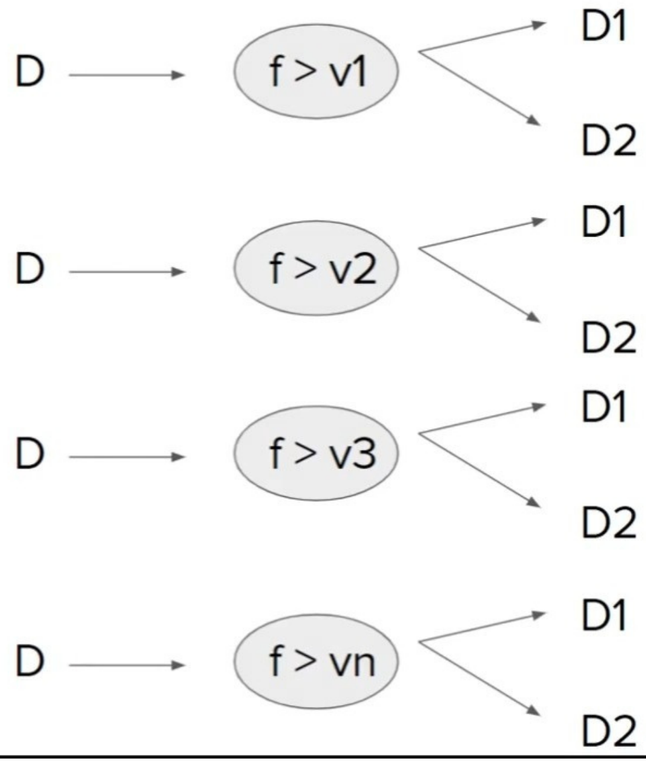
S No	User Rating	Downloaded
1	1.6	Yes

S No	User Rating	Downloaded
2	1.9	Yes
3	2.2	No
4	2.5	Yes
5	2.9	Yes
6	3.2	No
7	3.3	No
9	3.5	Yes
10	3.9	No
11	4.1	No
12	4.6	Yes
13	4.8	Yes

For Numerical Datasets

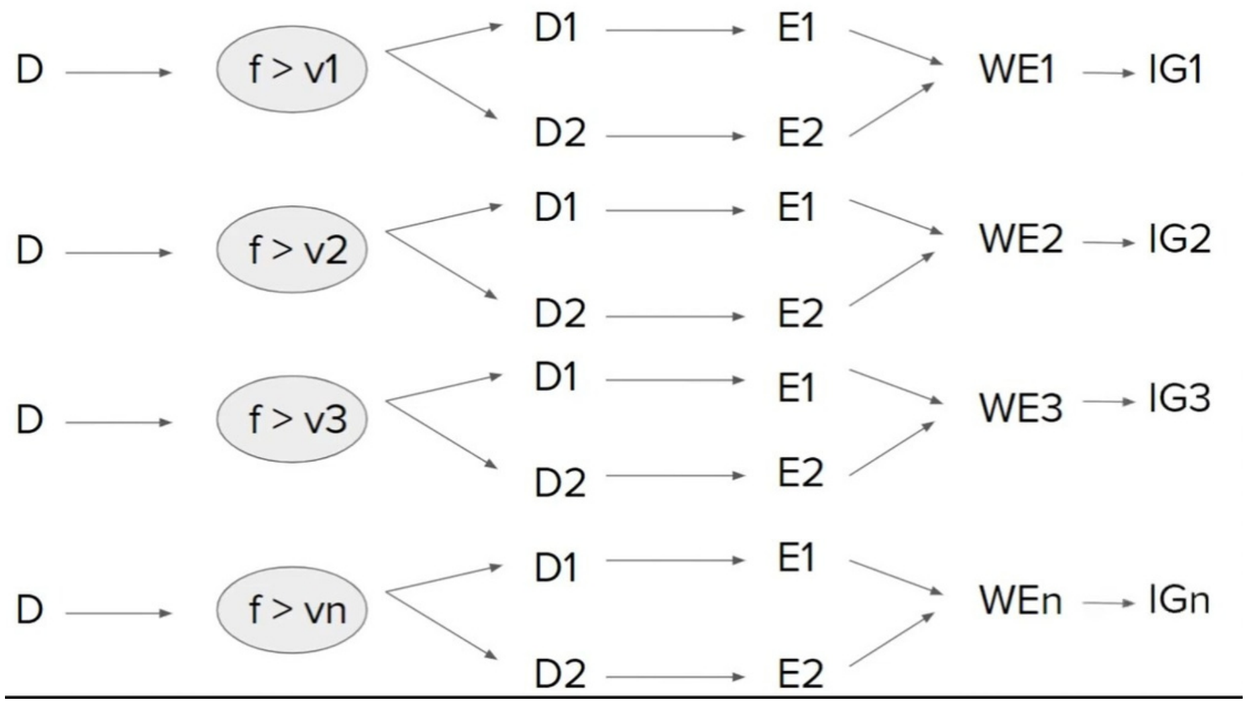
	User Rating	Downloaded
	1.6	Yes
	1.9	Yes
	2.2	No
	2.5	Yes
	2.9	Yes

S No	User Rating	Downloaded
6	3.2	No
7	3.3	No
9	3.5	Yes
10	3.9	No
11	4.1	No
12	4.6	Yes
13	4.8	Yes



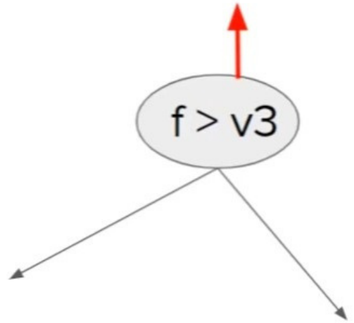
- Where D is the Dataset
- f is the column User Rating
- v1,v2,....vn are the values of various rows of User Rating

For Numerical Datasets



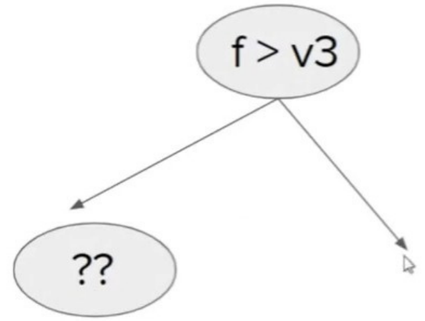
Step 5:

Max{ IG1, IG2, IG3....., IGn }



Step 6:

Do this recursively until you get all the leaf nodes



Numerical DataSets

Question

This entire thing looks computationally too expensive. Is is the right way of finding the splitting criteria?

Yes

Train ~~**Test**~~ **log(n)**

Because we are using only one time (in training data...)
We no need to again further....after deployment (test data) . So for Training data the time complexity is high but low for test that is log(n)

Gini Impurity

The gini impurity measures the frequency at which any element of the dataset will be mislabelled when it is randomly labeled.

It measures the impurity of the nodes

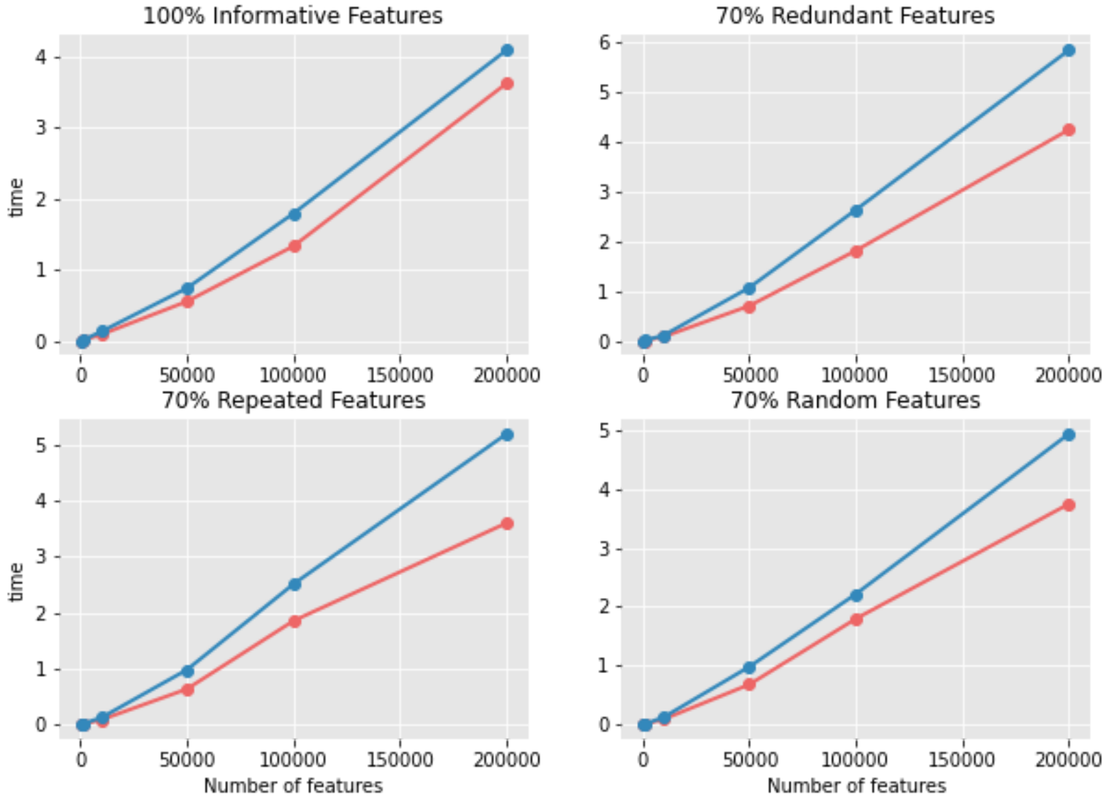
npurity is a measurement of the likelihood of an incorrect
cation of a new instance of a random variable, if that new instance
ndomly classified according to the distribution of class labels
he data set.

Gini impurity = 1 – Gini

$$Gini = (p_1^2 + p_2^2 + p_3^2 + \ldots + p_n^2)$$

Which is Better GINI vs Entropy

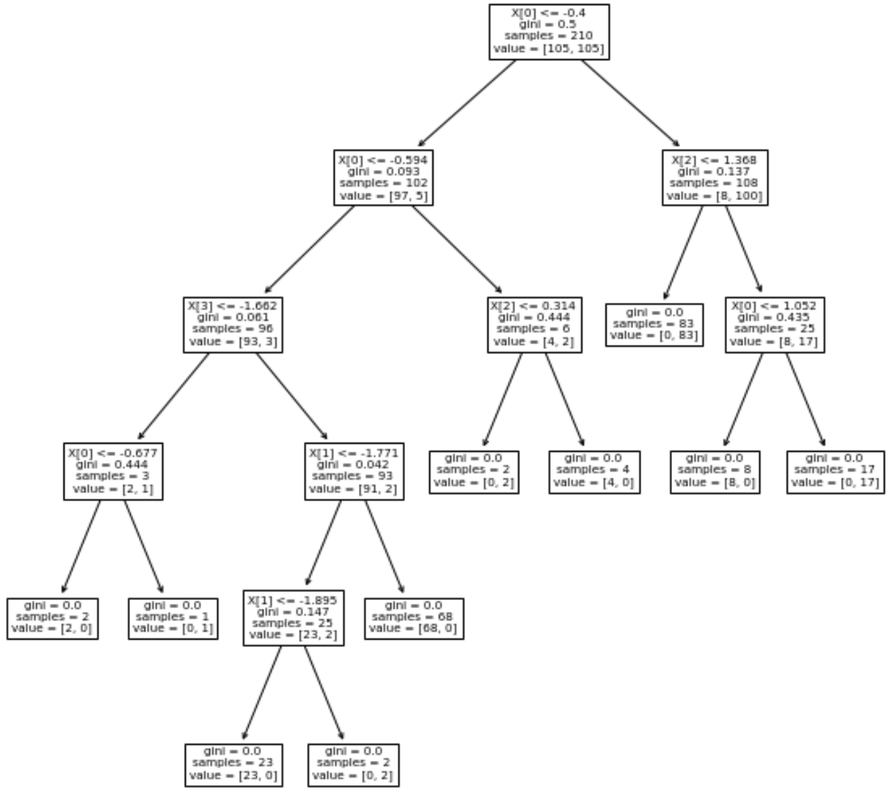
	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Informative	100%	30%	30%	30%
Redundant	0	70%	0	0
Repeated	0	0	70%	0
Random	0	0	0	70%

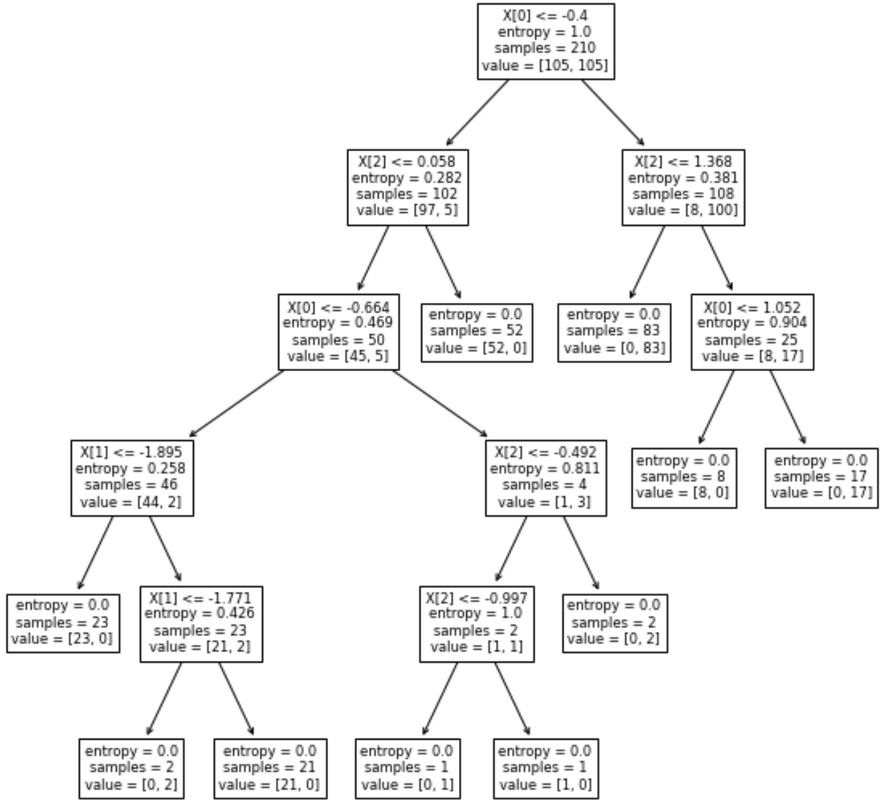


As can be seen, the training time when using the Entropy criterion is much higher. Furthermore, the impact of the nature of the features is quite noticeable. Redundant features are the most affected ones, making the training time 50% longer than when using informative features. Whereas, the use of random features or repeated features have a similar impact. The differences in training time are more noticeable in larger datasets.

Which is Better GINI vs Entropy

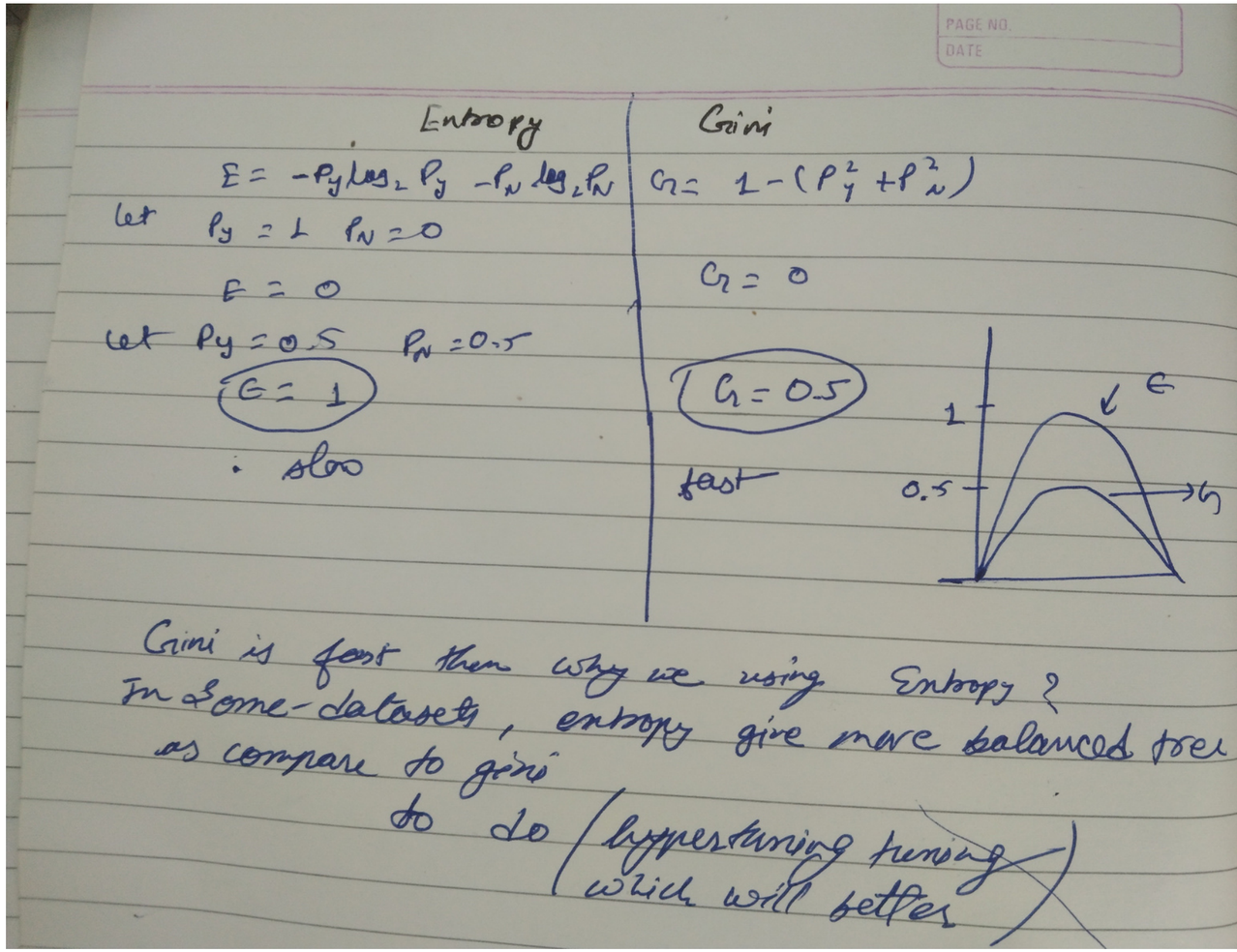
if we compare the structure of the trees, we can see that they are different. For that purpose, we have created a new synthetic dataset with 400 samples and 4 features. The obtained results are an F-Score of 0.93 for both criteria but the resulting tree is different. In addition, the first split in the two trees is the same as the branch on the right of the tree, however, the rest of the tree is different.
Although the trees are not equal, the obtained result is practically identical





Conclusion

The gini criterion is much faster because it is less computationally expensive.
The obtained results using the entropy criterion are slightly better.
Nevertheless, as the results are so similar, it does not seem to be worth the time invested in training when using the entropy criterion.



Hyperparameter Tuning

Hyperparameter tuning is searching the hyperparameter space for a set of values that will optimize your model architecture.

This is different from tuning your model parameters where you search your feature space that will best minimize a cost function

Criterion: gini and entropy

The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.

gini impurity-> $Gini: Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$

$Entropy: H(E) = -\sum_{j=1}^{c} p_j \log p_j$

Information gain uses the entropy measure as the impurity measure and splits a node such that it gives the most amount of information gain. Whereas Gini Impurity measures the divergences between the probability distributions of the target attribute’s values and splits a node such that it gives the least amount of impurity.

Decision Tree Hyperparameters

Max_Depth

- It indicates how deep the decision tree can be
- The deeper the tree, the more splits it has & it captures more information about the data
- However, in general a decision tree overfits for large depth values. The tree perfectly predicts all of the train data, however, it fails to generalize the findings for new data

min_samples_split

- An internal node will have further splits
- min-sample_split specifies the minimum number of samples required to split an internal node.
- We can either specify a number to denote the minimum number or a fraction to denote the percentage of samples in an internal node.

Decision Tree Hyperparameters

min_samples_leaf

- A leaf node is a node without any children (without any further splits)
- min_samples_leaf is the minimum number of samples required to be at a leaf node.
- This parameter is similar to min_samples_splits, however,this describe the minimum number of samples at the leafs, the base of the tree

max_features

- Max_features represents the number of features to consider when looking for the best split
- We can either specify a number to denote the max_features to consider while making a split
- We also have options such as sqrt,log2 ,None.

Decision Tree Hyperparameters

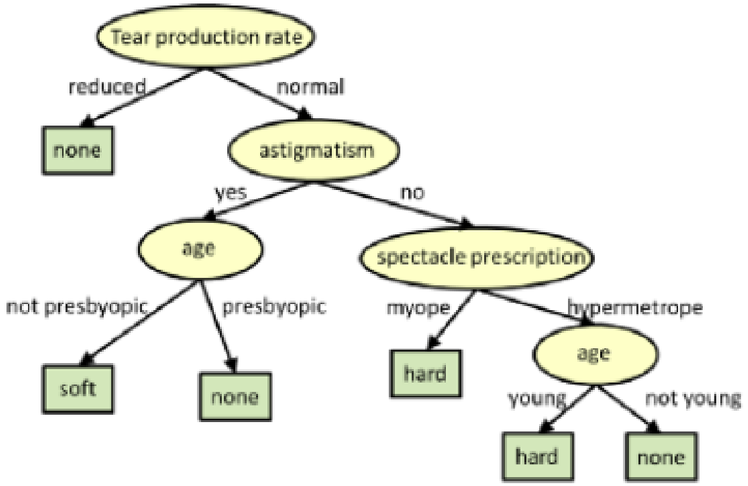
max_features

- Let us say the dimension of your data is 50 and the max_feature is 10, each time you need to find the split, you randomly select 10 features and use them to decide which one of the 10 is the best feature to use. When you go to the next node you select randomly another 10 and so on
- This mechanism is used to control overfitting. In fact, it is similar to the technique used in random forest, except in random forest we start with sampling also from the data and we generate multiple trees.

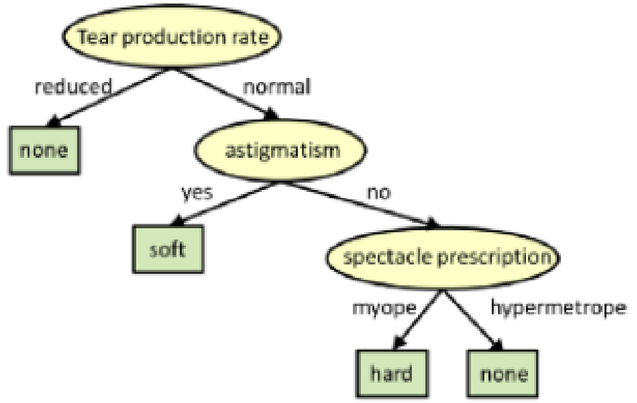
Pruning

In pruning, you trim off the branches of the tree, i.e., remove the decision nodes starting from the leaf node such that the overall accuracy is not disturbed. This is done by segregating the actual training set into two sets: training data set, D and validation data set, V. Prepare the decision tree using the segregated training data set, D. Then continue trimming the tree accordingly to optimize the accuracy of the validation data set, V.

There are mainly 2 ways for pruning:
(i) Pre-pruning – we can stop growing the tree earlier, which means we can prune/remove/cut a node if it has low importance while growing the tree.
(ii) Post-pruning – once our tree is built to its depth, we can start pruning the nodes based on their significance.



Original Tree



Pruned Tree

Advantages of Decision Tree (CART)

- **Easy to visualize and interpret:** Its graphical representation is very intuitive to understand and it does not require any knowledge of statistics to interpret it.
- **Useful in data exploration:** We can easily identify the most significant variable and the relation between variables with a decision tree. It can help us create new variables or put some features in one bucket.
- **Less data cleaning required**: It is fairly immune to outliers and missing data, hence less data cleaning is needed.
- **The data type is not a constraint:** It can handle both categorical and numerical data.
- Decision trees *implicitly perform variable screening or feature selection.*
- *Nonlinear relationships between parameters do not affect tree performance.*

analytic vidhya

Disadvantages of Decision Tree

- **Overfitting:** single decision tree tends to overfit the data which is solved by setting constraints on model parameters i.e. height of the tree and pruning
- **Not exact fit for continuous data:** It losses some of the information associated with numerical variables when it classifies them into different categories.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This is called variance, which needs to be lowered by methods like bagging and boosting.
- Greedy algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees, where the features and samples are randomly sampled with replacement.
- Decision tree learners create *biased trees if some classes dominate*. It is therefore recommended to balance the data set prior to fitting with the decision tree.

analytic vidhya

Regression Tree

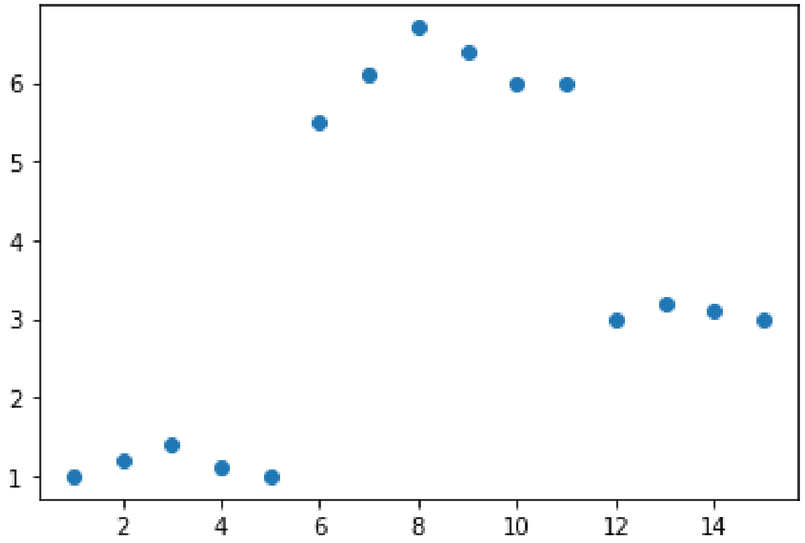
A regression tree is basically a decision tree that is used for the task of regression which can be used to predict continuous valued outputs instead of discrete outputs.

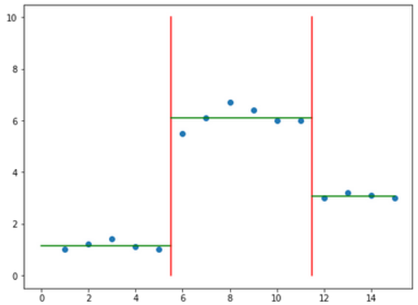
Mean Square Error

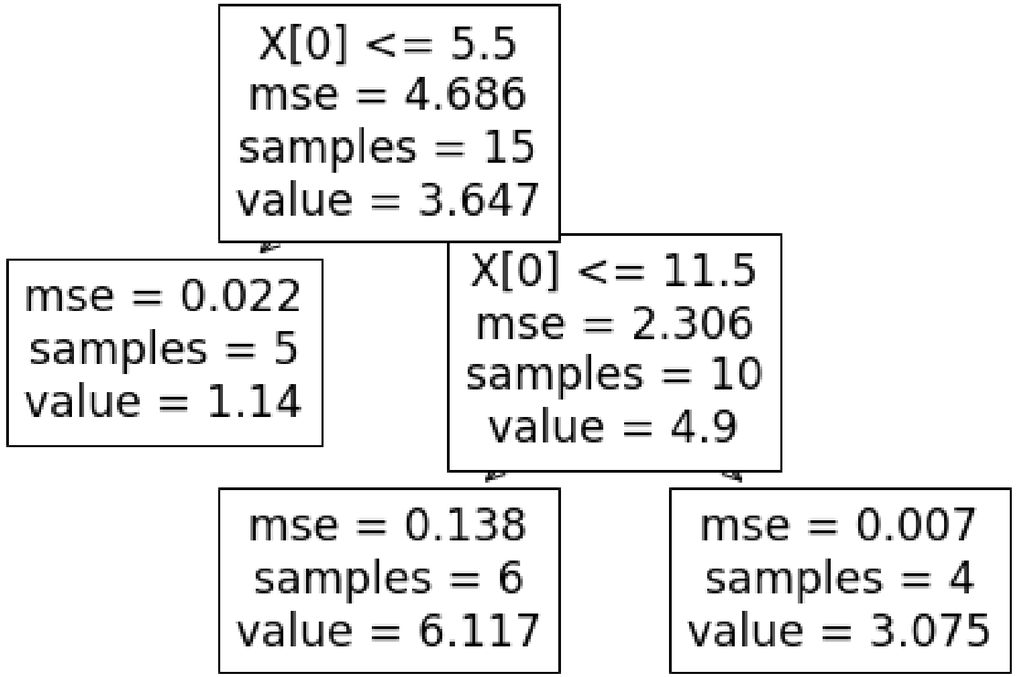
In Decision Trees for Classification, we saw how the tree asks right questions at the right node in order to give accurate and efficient classifications. The way this is done in Classification Trees is by using 2 measures , namely Entropy and Information Gain. But since we are predicting continuous variables, we cannot calculate the entropy and go through the same process. We need a different measure now. A measure that tells us how much our predictions deviate from the original target and that’s the entry-point of mean square error.

Regression Tree

Here Linear Regression will not work better , so we will use Regression Tree







we are selecting these points is by going through an iterative process of calculating mean square error for all the splits and choosing the split that has the least value for the *mse*. So, It only natural this works.

Regression Tree

What happens when there are multiple independent variables ?
Let us consider that there are 3 variables similar to the independent variable X.
At each node, All the 3 variables would go through the same process as what X went through in the last example. The data would be sorted based on the 3 variables separately.
The points that minimises the mse are calculated for all the 3 variables. out of the 3 variables and the points calculated for them, the one that has the least mse would be chosen.