

## **DSA GROUP PROJECT**

### **Topic Undertaken - DECISION TREE**



Group members

BT19CSE002 – Jatin Chhangani  
BT19CSE009 – Ruchika Pandharikar  
BT19CSE013 – Yashashree Ganthale  
BT19CSE031 – Aditi Vishwakarma  
BT19CSE033 – Samruddhi Selukar  
BT19CSE046 – Prabhu Satyam  
BT19CSE047 – Neha Kalbande  
BT19CSE053 – Nandini Kapoor  
BT19CSE056 – Hemanshu Chaudhari  
BT19CSE057 – Aditi Sahu

***Indian Institute of Information Technology, Nagpur***  
***Data Structures with Applications***  
***Sem 3 CSE***

## Decision Tree

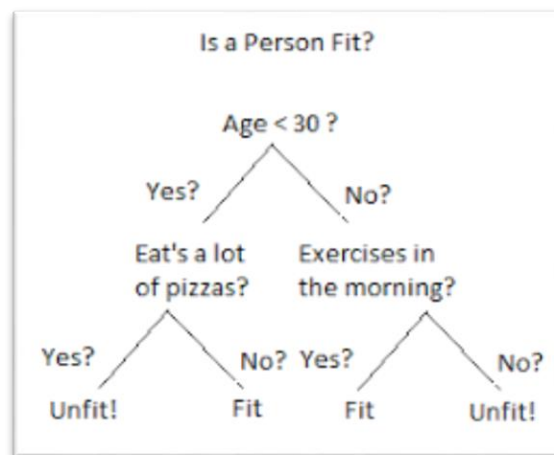
### **Introduction:**

#### 1. What is a Decision Tree?

A Decision tree is a type of classifier, a collection of nodes and vertices intended to form a conclusion based on available affiliated attribute values. They are self-learning trees which can be trained using a sample dataset, so that they'd be able to classify the object associated with a set of attribute values into labels that the objects most likely belongs to. These classify in a different manner than the Naïve Bayes classifier family of algorithms, which assume every pair of features being classified is independent of each other.

An internal node of a decision tree represents attributes, each branch represents a decision rule and each leaf node represents the outcomes. The topmost node in a decision tree is known as the root node and is also the first attribute based on which the classification progresses. It learns to partition based on the attribute values. It partitions the tree in a recursive manner called recursive partitioning. This flowchart-like structure helps you in decision making.

Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example, this approach is called a Top-Down approach. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case. This process is recursive and is repeated for every subtree rooted at the new nodes.



An example of Decision Tree

#### 2. Formal Definition:

A decision tree is a supervised machine learning tool used in classification problems to predict the class of an instance. We train the model using correctly labelled instances. With decision trees, we use our training data to build the attributes tests (internal nodes) and threshold values based on the information gain metric. The information gain metric chooses an attribute and threshold that maximizes information learned, which is calculated based on how well the attribute test splits the training data into two subsets each having all the same classification.

### **Properties:**

## Decision Tree

- Divide and conquer structure for sorting a training set into pure subsets.
- Leaves represent class labels and branches represent conjunctions of features that lead to these class labels.
- A decision tree model contains a key column, input columns, and at least one predictable (label/class) column.
- Decision trees use multiple algorithms to decide to split a node in two or more sub-nodes.
- The creation of sub-nodes increases the homogeneity of resultant sub-nodes i.e. purity of the node increases with respect to the target variable.
- Decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.
- The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria is different for classification and regression trees.

$$Variance = \frac{\sum (X - \mu)^2}{N}$$

- Variance:

- Entropy: Entropy is used to measure the impurity or randomness of a dataset

$$Entropy = - \sum_{i=1}^n p_i \log_2 p_i$$

- Gini: It is the probability of correctly labeling a randomly chosen element if it was randomly labeled according to the distribution of labels in the node.

$$Gini = \sum_{i=1}^n p_i^2$$

- Information Gain: Decision tree algorithm implementation uses information gain to decide which feature needs to be split in the next step. Information Gain is used for splitting the nodes when the target variable is categorical. It works on the concept of the entropy

$$Information\ Gain = 1 - Entropy$$

**Entropy and IG formulae explained:**

## Decision Tree

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). This partitioning introduces certain new criteria and terms to be aware of that enable in determining the point of division.

### 1. Information Gain:

Information gain (IG) measures how much “information” a feature gives us about the class.

- Information gain is the main key that is used by Decision Tree Algorithms to construct a Decision Tree.
- Decision Trees algorithm will always try to maximize Information gain.
- An attribute with highest Information gain will tested/split first.

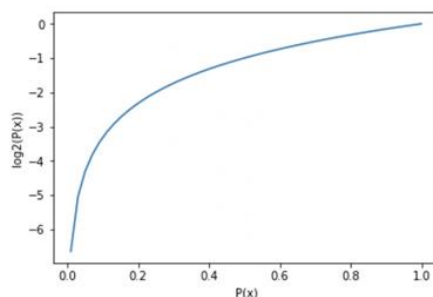
### 2. Entropy:

Entropy is used for calculating the purity of a node.

- Uncertainty
- Purity
- Information content

It is used to measure the impurity or randomness of a dataset. Imagine choosing a yellow ball from a box of just yellow balls (say 100 yellow balls). Then this box is said to have 0 entropy which implies 0 impurity or total purity.

Now, let's say 30 of these balls are replaced by red and 20 by blue. If we now draw another ball from the box, the probability of drawing a yellow ball will drop from 1.0 to 0.5. Since the impurity has increased, entropy has also increased while purity has decreased. Shannon's entropy model uses the logarithm function with base 2 ( $\log_2(P(x))$ ) to measure the entropy because as the probability  $P(x)$  of randomly drawing a yellow ball increases, the result approaches closer to binary logarithm 1 as shown in the graph below.



When a target feature contains more than one type of element (balls of different colors in a box), it is useful to sum up the entropies of each possible target value and weigh it by the probability of getting these values assuming a random draw. This finally leads us to the formal definition of Shannon's entropy which serves as the baseline for the information gain calculation:

$$Entropy = - \sum_{i=1}^n p_i \log_2 p_i$$

Logarithm of fractions gives a negative value and hence a '-' sign is used in entropy formula to negate these negative values.

$$\text{Information Gain} = 1 - \text{Entropy}$$

### **Operations on Decision Trees:**

#### **1. Creation (how to build tree):**

1. Select the first attribute of test dataset over which the set would split.
2. If a pure subset is obtained, no further splitting is performed for the pure value.
3. Grow the rest of the tree by splitting impure subsets based on new unused attributes with the help of "node impurity."

#### **2. Split (how to split node):** There are multiple ways of doing this, the suitable algorithm selection is based on type of target variables, which can be broadly divided into two categories:

1. Continuous Target Variable
  - Reduction in Variance
2. Categorical Target Variable
  - Information Gain
  - Gini Impurity

### **Reduction in Variance:**

Reduction in Variance is a method for splitting the node used when the target variable is continuous, i.e., regression problems. It is so-called because it uses variance as a measure for deciding the feature on which node is split into child nodes.

Variance is used for calculating the homogeneity of a node. If a node is entirely homogeneous, then the variance is zero. Here are the steps to split a decision tree using reduction in variance:

1. For each split, individually calculate the variance of each child node
2. Calculate the variance of each split as the weighted average variance of child nodes
3. Select the split with the lowest variance
4. Perform steps 1-3 until completely homogeneous nodes are achieved

### **Information Gain:**

It works on the concept of the entropy. Entropy is used for calculating the purity of a node. Lower the value of entropy from 1, the information gain is higher for the pure nodes with a maximum value of 1. ID3 algorithm uses entropy to calculate the homogeneity of a sample. Steps to split a decision tree using Information Gain:

1. For each split, individually calculate the entropy of each child nodes.

## Decision Tree

2. Calculate the entropy of each split as the weighted average entropy of child nodes.
3. Select the split with the lowest entropy or highest information gain.
4. Until you achieve homogeneous nodes, repeat steps 1-3.

### **Gini Impurity:**

It is the most popular and the easiest way to split a decision tree.

$$Gini\ Impurity = 1 - \sum_{i=1}^n p_i^2$$

Lower the Gini Impurity, higher is the homogeneity of the node.

Gini Impurity is preferred over Information Gain because it does not contain logarithms which are computationally intensive.

### **3. Termination (When to stop splitting):**

- No more input features.
- All examples are classified the same.
- Too few examples to make an informative split.

### ***Algorithm:***

1. Read the csv file as df.
2. Rename the last column from species to class.
3. `train_test_split(df, test_size):`
  1. `test_size = % of dataset being considered for training df is given rounded.`
  2. Randomly generate the `test_size` rows to test the data.
  3. Return the randomly generated data as well as the indices of the data.
4. Apply `decision_tree_algorithm(df, counter, min_sample, max_depth)` to create the decision tree :

1. `if counter==0`

Convert the data frame to 2D array named as 'data'.

2. Check purity or the length of data is `min_sample` or `counter==maxdept:`

`if true:`

`classification = classify_data (data)`

`return classification`

`else:`

1. Increment the counter.

2. `potential_splits = get_potential_splits(data)` # to generate all the potential splits condition.

3. `split_column, split_value = determine_best_split(data, potential_splits)`  
# to determine the best split among the potential split.

4. `data_below, data_above = split_data(data, split_column, split_value)`

## Decision Tree

```
# to split the data on the basis of the best split.
5. question = "{ } <= { }".format(feature_name, split_value)
6. sub_tree = {question: []}
7. # find answers (recursion)
   yes_answer = decision_tree_algorithm(data_below, counter, min_samples,
max_depth)
   no_answer = decision_tree_algorithm(data_above, counter, min_samples, max_depth)
8. if yes_answer == no_answer:
   sub_tree = yes_answer
9. else:
   sub_tree[question].append(yes_answer)
   sub_tree[question].append(no_answer)
   return sub_tree
5. calculate_accuracy(df,tree):
   Adding 2 more columns to the DataFrame df.
   First is classification which contents the result of the tree.
   Second is the Boolean value if the result value is matched with the actual value true or
   false

df["classification"] = df.apply(classify_example, axis=1, args=(tree,))
df["classification_correct"] = df["classification"] == df["class"]
accuracy = df["classification_correct"].mean()
return accuracy
```

### Helping Functions:

1. **check\_purity** : Checks if data is pure i.e check if there is some part of the data that lies completely separated from the other => definite boundry can be found for some labels such that dataset of one label does not interfere inthe dataset of the other label.
2. **classify\_data**: It not only classifies the data when it is pure but it also classifies the data when not completely pure based on the majority of species i.e which ever result appears most frequently.
3. **get\_potential\_splits**: Having the all the unique values in each column, the general idea is to consider all value present exactly in the middle of any 2 unique value as a potential split
4. **calculate\_entropy**: Entropy = summationof( $p(i) * (-\log_2(p(i)))$ ) where p is the probability of the unique individual element in the dataset of the selected column.
5. **calculate\_overall\_entropy**: overall entropy = summationof( $p(j) * \text{Entropy}$ )
6. **determine\_best\_split**: Calculating the overall entropy of every split of each columns and each value, and then try to find the minimum overall entropy of all, which will be the best split column and the best split value.
7. **split\_data**: The data is split w.r.t the respective column, and based on the appropriate split value into data below the value and above the line.

**Example (Simple for understanding, actual dataset used in example explained with the code):**

⇒ An oil company owns a land and wants to decide whether to drill to find oil.

## Decision Tree

They have 2 options:

Option 1: Directly determine whether to drill or sell the land.



Option 2: Do a seismic experiment, then determine whether to drill

- Seismic experiment cost: \$2 million
- If result is Favorable (40%), the chance of containing oil is 60%
- If result is Unfavorable (60%), the chance of containing oil is 10%

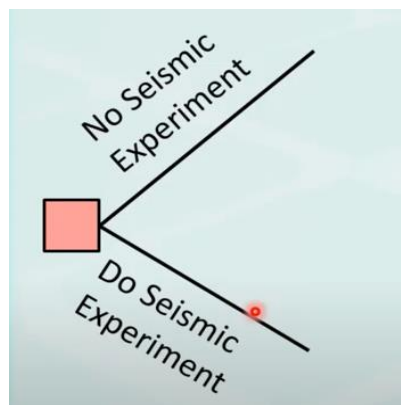
- 1) The drilling operation will cost \$5 million
- 2) If oil is found, the company will earn \$40 million
- 3) If the land is sold without drilling, the company will earn \$4 million

So, the problem arises that in order to maximize the company's earnings, What strategies should they follow?

Let's try to build a decision tree for this:

- Decision trees are used to make optimal decisions.
- A decision node (denoted by ) represents a time when a decision is made.
- An event node (denoted by ) represents a time when the outcomes occur.
- A branch of a decision tree is a terminal branch if no nodes come out of the branch.

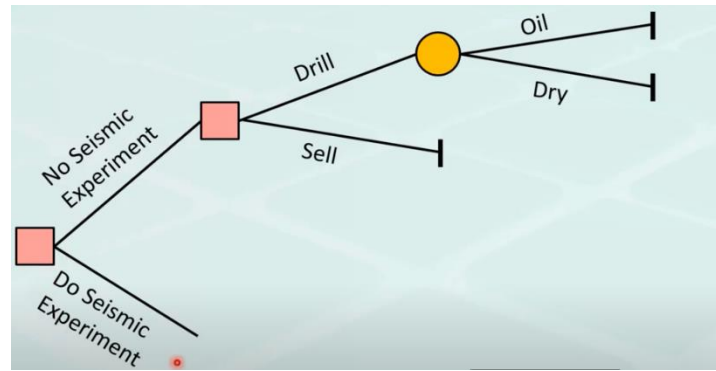
→ At the very beginning, they need to make a decision about whether they should make a seismic experiment or not



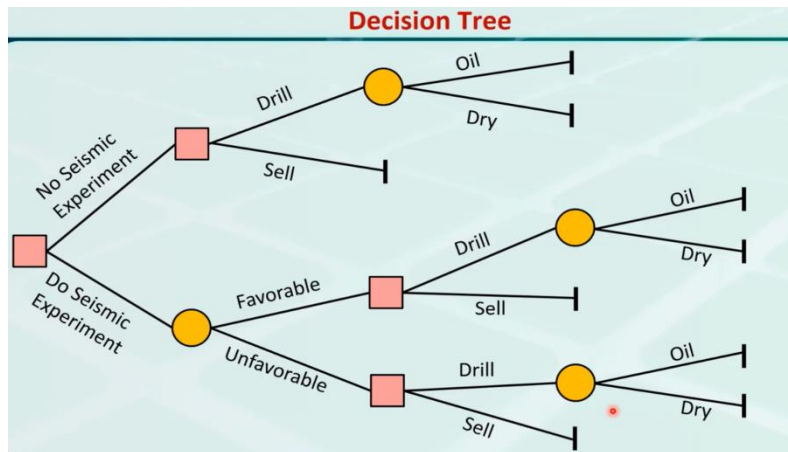
→ According to given conditions if company goes for no seismic experiment, then there comes again 2 decisions to drill or sell. After selling there is no further work so it is a terminal branch. If they go for drilling, there is need to observe outcome which are if land contains oil or it is dry.



## Decision Tree



→ Similarly going with the seismic experiment and then considering favorable and non-favorable conditions and further choosing drilling or selling, the final structure of our decision tree becomes:



→ Now let's put specific numbers on specific branches, as the cost of doing experiment and drilling are \$2 and \$5 respectively, it is written as negative and positive values shows the amount earned by company by selling

→ Calculate the final payoffs for each scenario:

- There is no cost involved with the no seismic experiment branch = 0
- Drilling operation cost = -5
- For oil found earning = +40

Thus, the final payoff is  $-5+40= 35$

Following the same rule, we calculate final payoffs for all as shown in figure below.

→ Put the probabilities on specific outcome branches as per the data given.

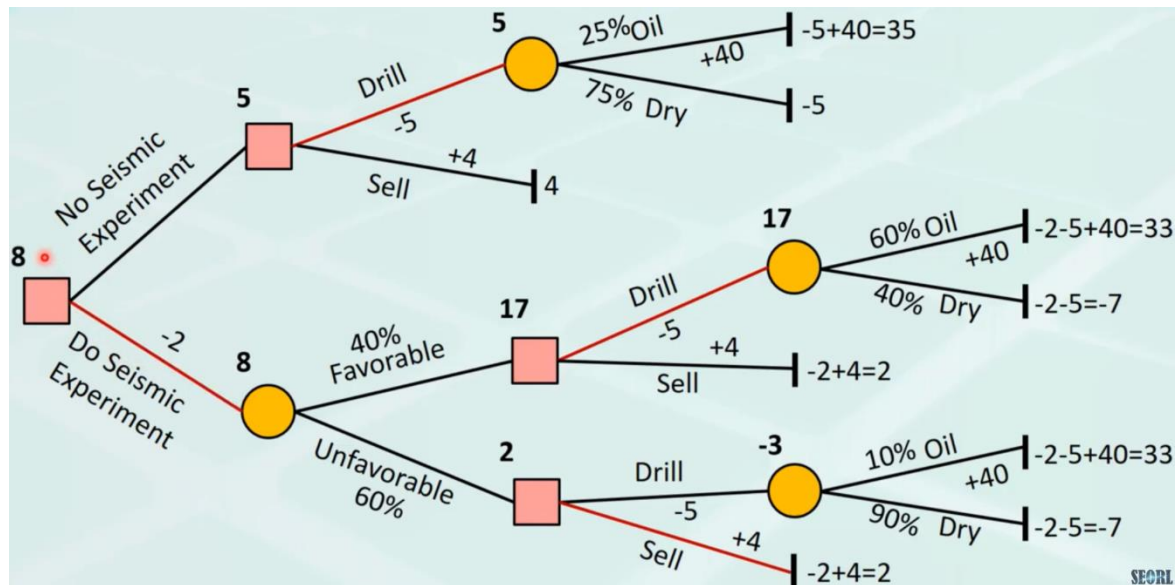
→ Finding expected payoff:

- 1) -For no seismic experiment drill =  $35*25\% + (-5) *75\% = 5$   
-For no seismic experiment sell= 4  
5 is greater than 4. So, Drill is a better decision in this case.
- 2) -For do experiment favorable drill=  $33*60\% + (-7) *40\% = 17$   
- For do experiment favorable sell= 2  
17 is greater than 2. So, Drill is a better decision in this case.

## Decision Tree

After calculating like above for each, the final payoff for seismic experiment is 8 and for non-seismic is 5.

Hence, **doing the seismic experiment is the better decision**



**Optimal strategy is:** Do the seismic experiment;  
If favorable, drill  
If unfavorable, sell  
Expected payoff is: \$8 million.

## ***Regression Tree with ID3 Algo (Sample Code to classify Iris dataset):***

The Iris dataset is in csv.

Code for the decision tree based on this dataset is in Jupyter Notebook written in python.

[code](#)

[dataset](#)

[Folder with dataset and code jupyter notebook](#)

The stepwise analysis has been given in the notebook so as to explain the algo with respect to the dataset used.

## ***Concurrency Control:***

- Time complexity of main algorithm (decision\_tree\_algorithm)
  - Best Case:  $O(MN \log(N))$
  - Worst Case:  $O(MN^2)$

Where M is number of features and N is the number of rows in Data

For Best Case:

If tree generated is balanced then the time complexity of tree will be  $O(\log(N))$

## Decision Tree

So, the total time complexity of the algorithm with splitting included will be  $O(MN \log(N))$

For Worst Case:

If tree generated is not balanced then the time complexity of the tree will be  $O(N)$

So, the total time Complexity of the algorithm with splitting included will be  $O(MN^2)$

### **Advantages:**

- Convenient determination: Decision tree is easy to understand and interpret. Help determine the expected values for different scenarios.
- Preventing bias: It enforces the consideration of all possible outcomes of a decision and traces each path to a conclusion.
- Not restricted by data: Decision trees are able to handle both continuous (through C4.5 algo by Ross Quinlan) and categorical variables.
- Most weighted fields: Decision trees provide a clear indication of which fields are most important for prediction or classification.
- Saves Time: Outcomes can be achieved faster through the use of a decision tree than manual classification.
- More Efficiency and Accuracy: supplies us with the most probable outcome of our problem with better accuracy than simpler probabilistic independent algorithms. Acceptable information we feed into the dataset improves the accuracy as the time progresses because with the historical records collected, the selections interpreted get higher and better.

### **Disadvantages:**

- They are unstable, meaning that a small change in the data can lead to a large change in the structure of the optimal decision tree.
- They are often relatively inaccurate in cases of multiple label cluster. Many other predictors perform better with similar data.
- Calculations can get very complex, particularly if many values are uncertain and/or if many outcomes are linked.
- For data including categorical variables with different number of levels, information gain in decision tree is biased in favor of those attributes with more levels.

### **Uses and Applications:**

- Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal.
- A popular tool in machine learning for classification of best matched label by attribute values.
- In systems analysis, trees are used mainly for identifying and organizing conditions and actions in a completely structured decision process. It is useful to distinguish between conditions and actions when drawing decision trees.
- The decision tree machine learning model can be used in data visualization, to explicitly present the decision process and its outcome.

## Decision Tree

### **References:**

[https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/trees/parallel\\_decision\\_tree.html](https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/trees/parallel_decision_tree.html)  
<https://www.youtube.com/watch?v=HQpWBcu5g4s&list=PLPOTBrypY74yzkgBMoXCqWQgwokErRkn5> -sebastian mantey introduction to ML  
[https://www.youtube.com/watch?v=Ud5XlhG8-gU&list=PLPOTBrypY74xS3WD0G\\_uzqPjCQfU6IRK-&index=4](https://www.youtube.com/watch?v=Ud5XlhG8-gU&list=PLPOTBrypY74xS3WD0G_uzqPjCQfU6IRK-&index=4) -sebastian mantey decision tree  
[https://www.youtube.com/watch?v=\\_XhOdSLIE5c](https://www.youtube.com/watch?v=_XhOdSLIE5c) -victor lavrenko ID3 algo  
[https://www.youtube.com/watch?v=AmCV4g7\\_-QM&list=PLBv09BD7ez\\_4temBw7vLA19p3tdQH6FYO](https://www.youtube.com/watch?v=AmCV4g7_-QM&list=PLBv09BD7ez_4temBw7vLA19p3tdQH6FYO) -victor lavrenko splitting attribute  
<https://www.geeksforgeeks.org/decision-tree/> -introduction to decision trees  
<https://stackoverflow.com/questions/40889344/decision-tree-using-continuous-variable> -continuous variable  
[https://en.wikipedia.org/wiki/ID3\\_algorithm](https://en.wikipedia.org/wiki/ID3_algorithm) -ID3 algo wiki  
<https://www.analyticsvidhya.com/blog/2020/06/4-ways-split-decision-tree/> -ways of splitting  
Classification and Regression Trees Leo Breiman  
C4.5 -Ross Quinlan

**THANK YOU...**