Data Mining Assignment (CSE3132)
Attribute Selection Algorithm

Srijit Roy, CSE-A, 3rd Year, 2351214

August 17, 2025

Introduction:-

Decision Trees are one of the most widely used supervised learning methods for classification and regression tasks. They split a dataset into smaller subsets based on attribute values, recursively forming a tree structure where each internal node represents an attribute test, each branch represents an outcome of the test, and each leaf node represents a class label or decision.

An important part of constructing a decision tree is the **Attribute Selection Algorithm**, which determines the best attribute to split the dataset at each step. Different impurity measures such as **Entropy, Information Gain, and Gini Index** are used for selecting the most informative attribute.

Formulas Used:-

Entropy

Entropy is a measurement of impurity or uncertainty in a dataset. It is represented as:-

$$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where p_i is the probability of class i.

Information Gain

Information Gain measures the reduction in entropy after splitting on an attribute. It is represented as:-

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

where S_v is the subset of S for which attribute A has value v.

Gini Index

The Gini Index measures impurity as:-

$$Gini(S) = 1 - \sum_{i=1}^{n} p_i^2$$

The weighted Gini Index for an attribute A is:-

$$GiniIndex(S, A) = \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Gini(S_v)$$

Python Implementation:-

The following Python program (Tree.py) implements the Decision Tree induction using Entropy and Gini Index as attribute selection measures, followed by a visualization of the tree using Graphviz:-

```
import numpy as np #np is the alias for numpy
import pandas as pd #pd is the alias for pandas
from collections import Counter
from graphviz import Digraph

Data=pd.read_csv("Dataset.csv") #reads the data from CSV file
features=list(Data.columns[1:-1])
target="Decision" #the last column is the independent column named
    Decision

def entropy(labels):
    c=Counter(labels) #counts the no of labels as Yes or No
    tot=len(labels) #counts total no of labels
    return -sum((count/tot)*np.log2(count/tot) for count in c.values())
        #Calculates total entropy

def info_gain(df,feature,target):
    base_entropy=entropy(df[target])
    values=df[feature].unique()
    wei_entropy = sum(
        (len(subset)/len(df))*entropy(subset[target]) #Calculates the
            individual info gains for each classes
        for v in values
        if len((subset:=df[df[feature]==v]))>0
    )
    return base_entropy - wei_entropy

def gini(labels):
    c=Counter(labels)
    tot=len(labels)
    return 1-sum((count/tot)**2 for count in c.values()) #Calculates
        each gini index

def gini_index(df,feature,target):
    values=df[feature].unique()
    wei_gini = sum(
        (len( subset )/len(df)) * gini ( subset [ target ])
        for v in values
```

```
        if len (( subset := df[df[ feature ] == v])) > 0
    )
    return wei_gini

def build_decision_tree(df,features,target,method):
     labels=df[target].tolist()
     if labels.count(labels[0])==len(labels):
         return labels[0] # Pure leaf
     if not features:
         return Counter(labels).most_common(1)[0][0]
     if method=="entropy":
        gains={f:info_gain(df,f,target) for f in features}
        best_feature=max(gains,key=gains.get)
     else:
        ginis={f:gini_index(df,f,target) for f in features}
        best_feature = min (ginis,key=ginis.get)

     d_tree={best_feature:{}}
     for v in df[ best_feature ].unique():
        subset=df[df[best_feature]==v].drop(columns=[best_feature])
        sub_features=[f for f in features if f!=best_feature]
        d_tree[best_feature][v]=build_decision_tree(subset,sub_features
           ,target,method)
     return d_tree

no_of_nodes=0 #counter for no of nodes
def node_id():
    global no_of_nodes
    no_of_nodes+=1
    return f"node{no_of_nodes}"

def visualize_tree(tree,graph=None,parent=None,edge_label=""):
    if graph is None:
        graph=Digraph(format="png")
        graph.attr(rankdir="TB",splines="polyline") # Better layout
        graph.attr("node",fontname="Helvetica")
    if isinstance(tree,dict):
        for feature,branches in tree.items():
            id_of_node=node_id()
            graph.node(id_of_node,feature,shape="box",style="rounded,
               filled",color="lightblue")
            if parent:
                graph.edge(parent,id_of_node,label=edge_label)
            for value,subtree in branches.items():
                visualize_tree(subtree,graph,id_of_node,str(value))
    else:
        id_of_leaf=node_id()
        graph.node(id_of_leaf,str(tree),shape="ellipse",style="filled",
           color ="lightgreen")
        if parent:
            graph.edge(parent,id_of_leaf,label=edge_label)
    return graph

tree_entropy=build_decision_tree(Data,features,target,method="entropy")
graph_entropy=visualize_tree(tree_entropy)
graph_entropy.render("tree_entropy",view=True )
no_of_nodes = 0
tree_gini=build_decision_tree(Data,features,target,method="gini")
```

```
graph_gini=visualize_tree(tree_gini)
graph_gini.render("tree_gini",view=True)
```

Listing 1: Decision Tree Implementation (Tree.py)

Provided Dataset:-

The dataset used for training the decision tree is shown below:

Weather	Parents	Financial Condition	Decision
Sunny	Yes	Poor	Cinema
Sunny	No	Rich	Play Tennis
Windy	Yes	Poor	Cinema
Windy	No	Poor	Cinema
Windy	No	Rich	Shopping
Rainy	Yes	Poor	Cinema
Rainy	No	Poor	Stay in
Rainy	No	Rich	Shopping

Induced Graphs:-

The constructed decision trees using different attribute selection measures are as follows:

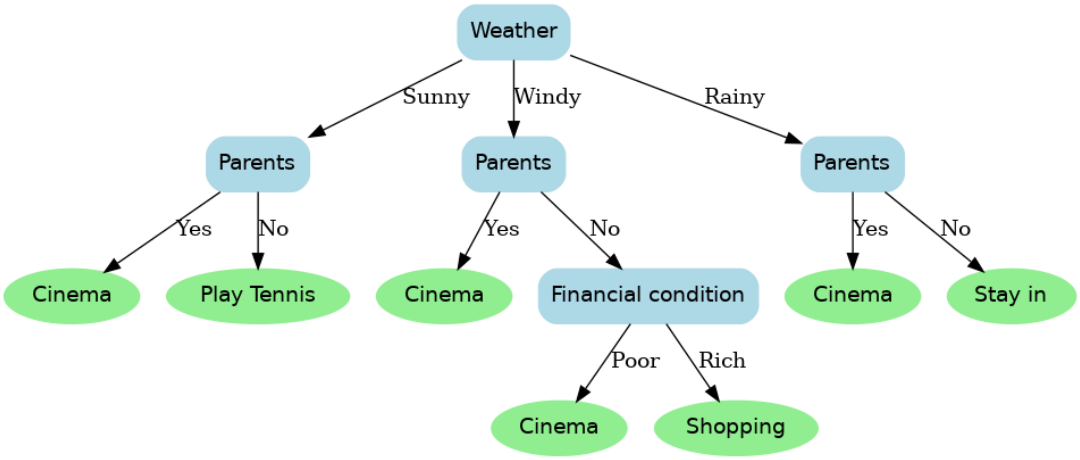

Figure 1: Decision Tree using Entropy (Information Gain)

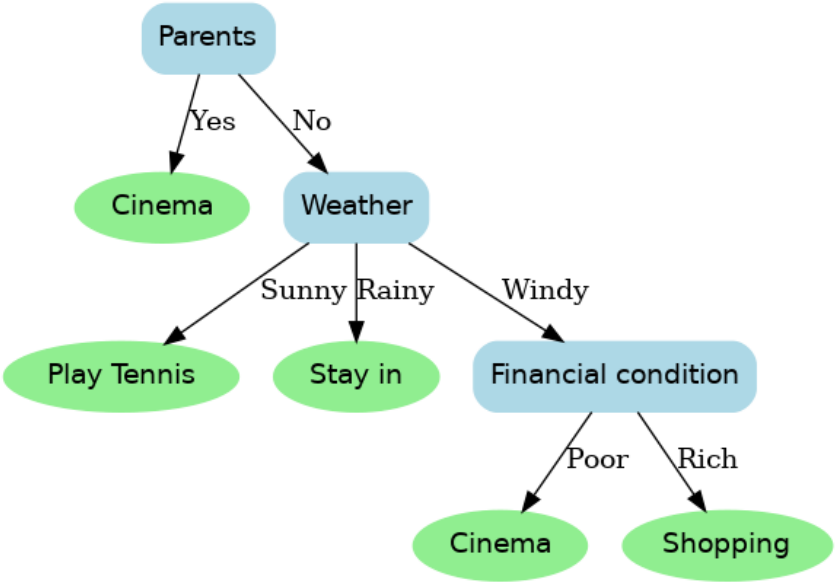

Figure 2: Decision Tree using Gini Index