Decision Tree Classifier

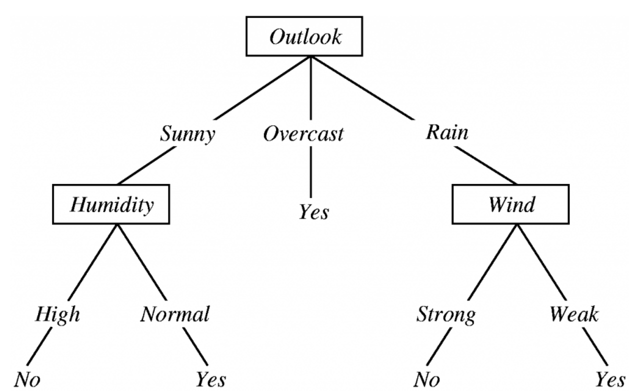
January 2022

1 Machine Learning Lab#5

1.1 Aim: Implement Decision Tree Classifier Algorithm on the given dataset

2 Introduction

Decision tree learning is a supervised learning method in which the learned function is represented by a decision tree. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label.



To understand the concept of Decision Tree consider the above example. Let's say you want to predict whether Saturday mornings are suitable for playing tennis, given information like weather, temperature, outlook and wind. The decision nodes are the questions "Whether the wind is strong or weak?", "Whether it is rainy outside?","Is the Humidity high or low?" And the leaf node represents outcome either 'Yes' or 'No'.

In general the decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances. Each path from the tree root to a leaf corresponds to a conjunction attribute tests, and the leaf itself to a disjunction of these conjunctions.

For the decision tree construction, the primary question that comes in mind is "which attribute should be tested at the root node?". To answer this question we have various attribute selection methods where each attribute is evaluated using a statistical test to determine how well it alone classifies the training examples. The best attribute is selected and used as the test at the root node of the tree. Descendent nodes are created for this root node for each possible value of this attribute and the training examples are sorted to appropriate descendent nodes. The entire process is then repeated using the training examples associated with each descendent node.

2.1 Attribute Selection Methods

2.1.1 Information Gain

In order to define Information gain precisely, we begin by defining a measure commonly used in Information Theory called Entropy. Entropy represents the impurity of an arbitrary collection of examples.

Impurity is the degree of randomness; it tells how random our data is. A pure sub-split means all the data samples belong to the same class.

If the target attribute can take c different values, then the entropy of S relative to this c-wise classification is defined as,

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

where p_i is the proportion of S belonging to class i.

To illustrate, suppose S is a collection of 14 examples of some boolean concept, including 9 positive and 5 negative examples. Then the entropy of S is given by,

$$Entropy([9+,5-]) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$$

$$= 0.940$$

Information gain is simply the reduction in entropy caused by partitioning the examples according to a particular attribute.

The information gain, Gain(S,A) of an Attribute A_i relative to the collection of examples S, is defined as,

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

where Values(A) is a set of all possible values for Attribute A and S_v is the subset of S for which attribute A has value v. The first term in the equation is the Entropy of the entire system. The second term is the expected value of entropy after S is partitioned using attribute A. Information gain represents the expected reduction in entropy caused by knowing the value of attribute A.

Information gain is precisely the measure used by ID3 to select the best attribute at each step of growing tree.

To illustrate the operation of attribute selection, let us consider the following example. Here, the target attribute Play Tennis, which can have yes or no for different Saturday mornings, is to be predicted based on other attributes of the morning in question.

According to the information gain measure, the Outlook attribute provides the best prediction of the target attribute, Play Tennis, over the training example. Therefore, Outlook is chosen as the decision attribute for the root node and branches are created below for each of its possible values (Sunny, Rainy and Overcast).

The process of selecting a new attribute and partitioning the training examples is now repeated for each non terminal descendent node, this time using only the training examples associated with that node.

Example For Information Gain:

Dataset:

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Rainy	Hot	High	False	No
2	Rainy	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Sunny	Mild	High	False	Yes
5	Sunny	Cool	Normal	False	Yes
6	Sunny	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Rainy	Mild	High	False	No
9	Rainy	Cool	Normal	False	Yes
10	Sunny	Mild	Normal	False	Yes
11	Rainy	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Sunny	Mild	High	True	No

[ ]:
```
Here the dataset is a collection of 14 tuples. Of these 14 tuples [9+,5-].

Entropy of the system is 0.940.
```

For example, Suppose we are splitting the tuples by the attribute outlook.

→ Attribute outlook contains,

Overcast → 4+, 0−
Rainy → 2+, 3−
Sunny → 3+, 2−

So, the Gain(S, A) where A = Outlook is given by,

$$\text{Gain}(S, \text{outlook}) = \text{Entropy}(S) - \sum_{V \in \{\text{Sunny, Rainy, overcast}\}} \frac{|S_V|}{|S|} \text{Entropy}(S_V)$$

$$= \text{Entropy}(S) - \left[\frac{4}{14}\left[\text{Entropy}(S_{overcast})\right]\right] - \left[\frac{5}{14}\left[\text{Entropy}(S_{rainy})\right]\right] - \left[\frac{5}{14}\left[\text{Entropy}(S_{sunny})\right]\right]$$

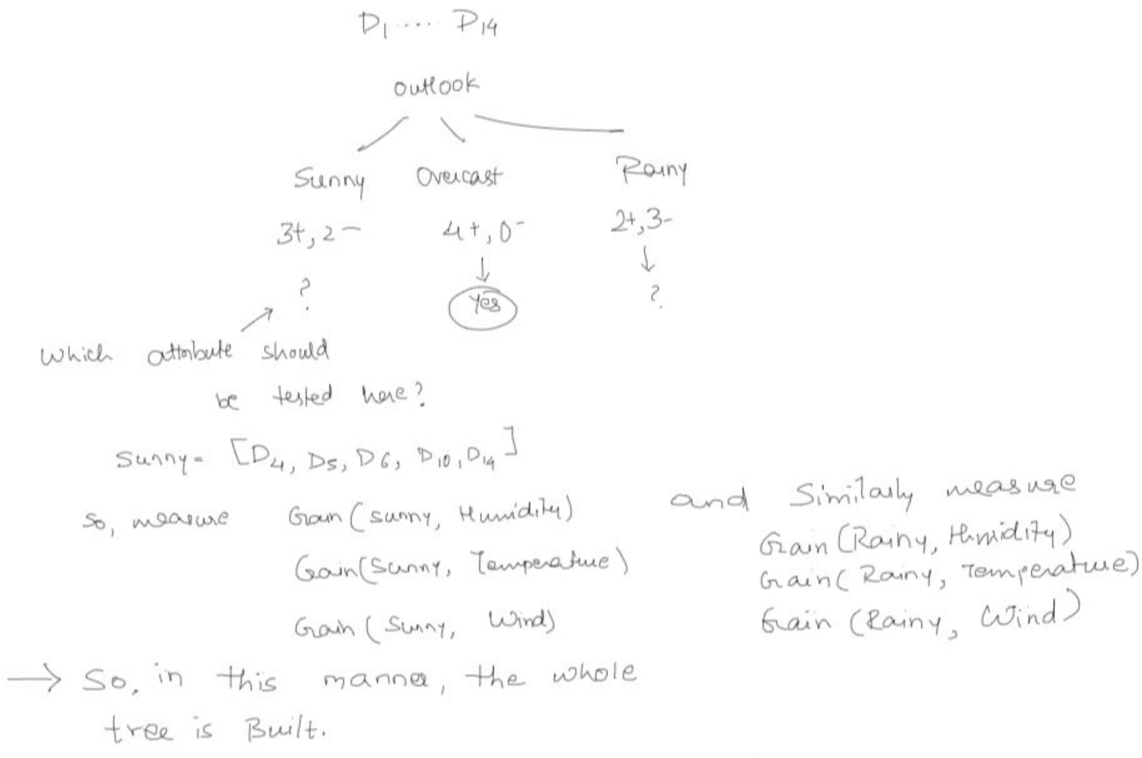
$$= \text{Entropy}(S) - \left[\frac{4}{14}\left[-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right]\right] - \left[\frac{5}{14}\left[-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right]\right] - \left[\frac{5}{14}\left[\left(-\frac{3}{5}\log_2\frac{3}{5}\right) - \frac{2}{5}\log_2\frac{2}{5}\right]\right]$$

[log0 is considered 0 here]

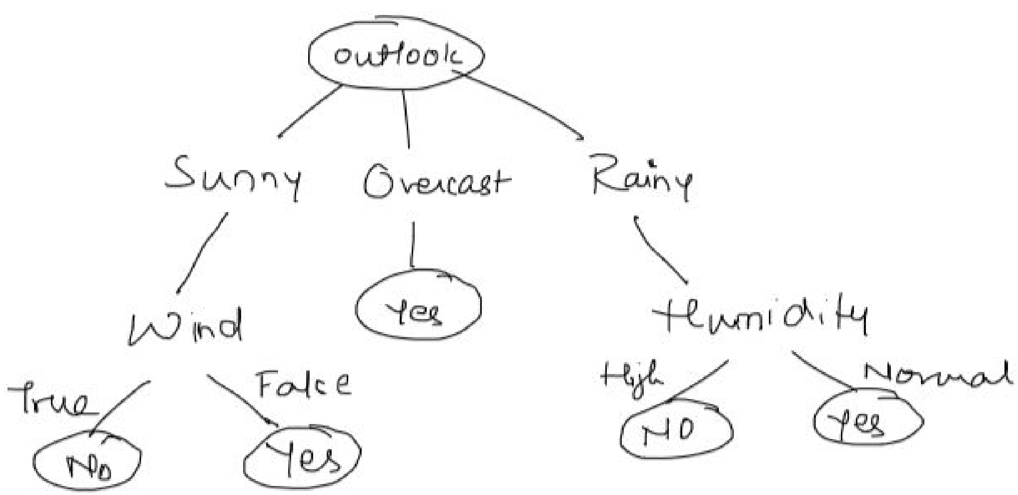
Gain(S, outlook) = 0.246

[ ]:
```
Similarly,

Gain(S,Humidity)= 0.151
Gain(S, Wind) = 0.048
Gain(S, Temperature)= 0.029
```




```
[ ]: So, the final tree constructed based on the above calculation is as
     follows:
```



2.1.2 Gini Index

Another decision tree algorithm CART (Classification and Regression Tree) uses the Gini method to create split points.

$$Gini(D) = 1 - \sum_{i=1}^{c} {p_i}^2$$

where p_i is the probability of tuple D belonging to a particular class. The optimal split is chosen by the attribute which has minimum gini index.

The Gini Index considers a binary split for each attribute. One can compute a weighted sum of the impurity of each partition. If a binary split on attribute A partitions data D into D1 and D2, the Gini index of D is:

$$Gini_A(D) = \frac{|D1|}{|D|}Gini(D1) + \frac{|D2|}{|D|}Gini(D2)$$

The Gini index considers a binary split for each attribute. Let's first consider the case where A is a discrete-valued attribute having v distinct values, $a_1, a_2, ..., a_v$, occurring in D. To determine the best binary split on A, we examine all the possible subsets that can be formed using known values of A. If A has v possible values, then there are 2^v possible subsets. For example, if income has three possible values, namely low, medium, high, then the possible subsets are low, medium, high, low, medium, low, high, medium, high, low, medium, high and .We exclude the power set, low, medium, high, and the empty set from consideration since, conceptually, they do not represent a split. Therefore, there are $2^v - 2$ possible ways to form two partitions of the data, D, based on a binary split on A.

For a discrete-valued attribute, the subset that gives the minimum Gini index for that attribute is selected as its splitting subset. For continuous-valued attributes, each possible split-point must be considered. The point giving the minimum Gini index for a given (continuous-valued) attribute is taken as the split-point of that attribute.

Gini Index Example:

→ Consider the following Dataset.

Age	Income	student	credit_rating	Buys_computer
Youth	high	No	fair	no
Youth	high	No	excellent	No
Middle-aged	high	No	fair	Yes
Senior	medium	No	fair	Yes
Senior	low	Yes	fair	Yes
Senior	low	Yes	excellent	No
Middle-aged	low	Yes	excellent	Yes
youth	medium	No	fair	No
Youth	low	Yes	fair	Yes
senior	medium	Yes	fair	Yes
Youth	medium	Yes	excellent	Yes
Middle-aged	medium	No	excellent	Yes
Middle-aged	high	Yes	fair	Yes
Senior	medium	No	excellent	No

→ Gini Index of Class Attribute
Total tuples = 14 → 9 → Yes, 5 → No

$$gini(D) = 1 - \left[\left(\tfrac{9}{14}\right)^2 + \left(\tfrac{5}{14}\right)^2\right] = 0.459$$

→ Now we need to compute Gini Index of each attribute (age, income, student and credit rating)

→ Let us consider age: {youth, middle-aged, senior}

Now consider each possible splitting subset: {youth, middle_age}, {middle_age, senior} ,{youth,senior} ,{youth},{middle_age},{senior}.

→ So if split is performed on {{youth, middle-aged}
→ 9 samples in [youth, middle-aged]
and 5 samples in senior

So, $Gini_A(D) = \frac{D_1}{D} gini(D_1) + \frac{D_2}{D} gini(D_2)$

So, $Gini_A(D) = \frac{9}{14}\left[1 - \left(\frac{6}{9}\right)^2 - \left(\frac{3}{9}\right)^2\right] + \frac{5}{14}\left[1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right]$

$Gini_A(D) = 0.4571$ where age ∈ (senior) & (youth, middle-aged)

→ Out of the 9 samples in (youth and middle-aged)
→ 6 are positive & 3 are negative.

→ similarly, out 5 samples in senior, 3 are positive & 2 are negative

⇒ Similarly we calculate the Gini Index for other splits.

```
So, Gini Index for {youth, senior} and {middle_age} = 0.3571
```

```
Gini Index for {middle_age, senior} and {youth} = 0.3936
```

So, here we will choose {youth, senior} and {middle_age} as splitting criteria if we perform splitting based on attribute age.

→ Again we will repeat this process for next attribute. that is income, student & credit_rating.

→ Out of all the attributes whichever attribute gives maximum reduction in Impurity will be choosen.

DT Weather Entropy

Decision Tree Classifier

January 2022

Aim: Implement Decsion Tree classifier

- Implement Decision Tree classifier using scikit learn library
- Test the classifier for Weather dataset

Step 1: Import necessary libraries.

```
[ ]: from sklearn import preprocessing
     from sklearn.tree import DecisionTreeClassifier
```

Step 2: Prepare dataset.

```
[ ]: #Predictor variables
     Outlook = ['Rainy', 'Rainy', 'Overcast', 'Sunny', 'Sunny', 'Sunny',␣
      ↪'Overcast',
                 'Rainy', 'Rainy', 'Sunny', 'Rainy','Overcast', 'Overcast',␣
      ↪'Sunny']
     Temperature = ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool',
                    'Mild', 'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild']
     Humidity = ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal',
                 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal',␣
      ↪'High']
     Wind = ['False', 'True', 'False', 'False', 'False', 'True', 'True',
                 'False', 'False', 'False', 'True', 'True', 'False', 'True']

     #Class Label:
     Play = ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No',
     'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
```

Step 3: Digitize the data set using encoding

```
[ ]: #creating labelEncoder
     le = preprocessing.LabelEncoder()

     # Converting string labels into numbers.
     Outlook_encoded = le.fit_transform(Outlook)
```

```
Outlook_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Outllok mapping:",Outlook_name_mapping)

Temperature_encoded = le.fit_transform(Temperature)
Temperature_name_mapping = dict(zip(le.classes_, le.transform(le.
 ↪classes_)))
print("Temperature mapping:",Temperature_name_mapping)

Humidity_encoded = le.fit_transform(Humidity)
Humidity_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Humidity mapping:",Humidity_name_mapping)

Wind_encoded = le.fit_transform(Wind)
Wind_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Wind mapping:",Wind_name_mapping)

Play_encoded = le.fit_transform(Play)
Play_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Play mapping:",Play_name_mapping)

print("\n\n")
print("Weather:" ,Outlook_encoded)
print("Temerature:" ,Temperature_encoded)
print("Humidity:" ,Humidity_encoded)
print("Wind:" ,Wind_encoded)
print("Play:" ,Play_encoded)
```

```
Outllok mapping: {'Overcast': 0, 'Rainy': 1, 'Sunny': 2}
Temperature mapping: {'Cool': 0, 'Hot': 1, 'Mild': 2}
Humidity mapping: {'High': 0, 'Normal': 1}
Wind mapping: {'False': 0, 'True': 1}
Play mapping: {'No': 0, 'Yes': 1}



Weather: [1 1 0 2 2 2 0 1 1 2 1 0 0 2]
Temerature: [1 1 1 2 0 0 0 2 0 2 2 2 1 2]
Humidity: [0 0 0 0 1 1 1 0 1 1 1 0 1 0]
Wind: [0 1 0 0 0 1 1 0 0 0 1 1 0 1]
Play: [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

Step 4: Merge different features to prepare dataset

[ ]:

```

```

Step 5: Train 'Create and Train DecisionTreeClassifier'

```
[ ]: #Create a Decision Tree Classifier (using Entropy)


     # Train the model using the training sets
```

Step 6: Predict Output for new data

```
[ ]: #Predict Output
```

Step 7: Display Decsion Tree Created

- This step requires graphviz and tkinter packages installed

```
[ ]: from sklearn.tree import export_graphviz
     #export_graphviz(clf_entropy,out_file='tree_entropy.dot',
      #              ␣
      ↪feature_names=['outlook','temperature','humidity','wind'],
       #              class_names=['play_no','play_yes'],
        #             filled=True)

     # Convert to png
     #from subprocess import call
     #call(['dot', '-Tpng', 'tree_entropy.dot', '-o', 'tree_entropy.png',␣
      ↪'-Gdpi=600'])

     # Display in python
     #import matplotlib.pyplot as plt
     #plt.figure(figsize = (14, 18))
     #plt.imshow(plt.imread('tree_entropy.png'))
     #plt.axis('off');
     #plt.show();
```