Lecture 12 – C4.5

Mr. Yousif G. Arshak
University of Zakho
Computer Science Department
yousif.arshak@uoz.edu.krd

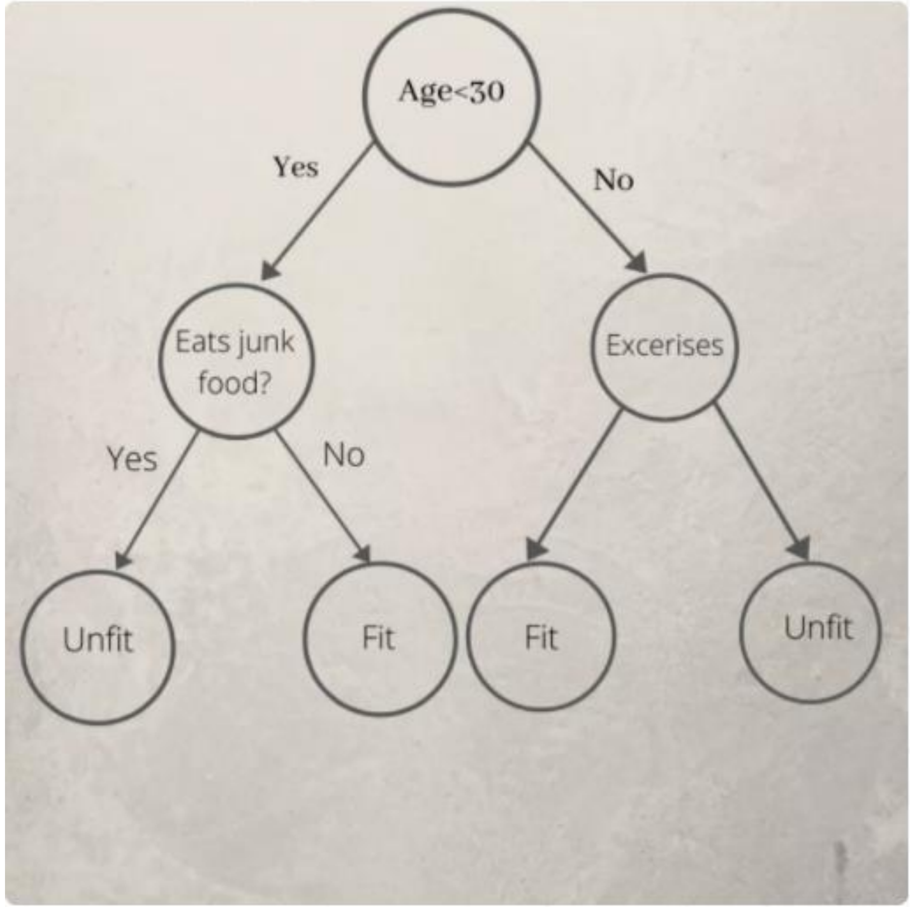
OUTLINES

- C4.5
- Introduction
- C4.5 Theory
- Implementation of C4.5 in Python

Introduction

- Decision Trees are the easiest and most popularly used supervised machine learning algorithm for making a prediction.
- The decision trees algorithm is used for regression as well as for classification problems. It is very easy to read and understand.
- Decision Trees are flowchart-like tree structures of all the possible solutions to a decision, based on certain conditions. It is called a decision tree as it starts from a root and then branches off to a number of decisions just like a tree.
- The tree starts from the root node where the most important attribute is placed. The branches represent a part of entire decision and each leaf node holds the outcome of the decision.

Decision Tree for predicting if a person is fit or unfit.



Attribute Selection Measure

- Attribute selection measure is a technique used for the selecting best attribute for discrimination among tuples. It gives rank to each attribute and the best attribute is selected as splitting criterion.
- The most popular methods of selection are:
    1. Entropy
    2. Information Gain
    3. Gain Ratio
    4. Gini Index

Entropy

- Entropy is the randomness in the information being processed.
- It measures the purity of the split.
- It ranges between 0 to 1. 1 means that it is a completely impure subset.

$$H(s) = -P_{(+)} + \log_2 P_{(+)} - P_{(-)} \log_2 P_{(-)}$$

- **Here, P(+) /P(-) = % of +ve class / % of -ve class**

Example on Entropy

- If there are total 100 instances in our class in which 30 are positive and 70 are negative then,
- P(+) = 3/10 and P(-) = 7/10
- H(s)= -3/10 * log2 (3/10) - 7/10 * log2 (7/10) ≈ 0.88

Information Gain

- nformation gain is a decrease in entropy. Decision trees make use of information gain and entropy to determine which feature to split into nodes to get closer to predicting the target and also to determine when to stop splitting.

$$Gain(S,A) = H(s) - \sum veValues\ (A) \frac{|s_v|}{|s|} \cdot H(S_v)$$

- Here, S is a set of instances , A is an attribute and S_v is the subset of S .

Example Information Gain

Sno.	Monthy income >1000$	TV at home
1	True	Yes
2	True	Yes
3	False	No
4	True	No
5	False	Yes
6	False	No
7	True	Yes
8	False	No
9	True	No
10	True	Yes

Possession of TV at home against monthly income

- For overall data, **Yes** value is present **5 times** and **No** value is present **5 times**. So,

$$H(s) = -[(5/10) * \log2 (5/10) + (5/10) * \log2 (5/10)] = 1$$

- To analyze **True values** now. **Yes** is present **4 times** and **No** is present **2 times.**

$$H(s) = -[(4/6) * \log2 (4/6) + (2/6) * \log2 (2/6)] = 0.917$$

For **False values**,

$H(s)= - [(3/4) * \log2 (3/4) + (1/4) * \log2 (1/4)] = 0.811$

Net Entropy = $(6/10) * 0.917 + (4/10) * 0.811 = 0.874$

Total Reduction = $1- 0.874 = 0.126$

- This value (0.126) is called information gain.

Gain Ratio

- Gini index is also type of criterion that helps us to calculate information gain. It measures the impurity of the node and is calculated for binary values only.
  - GR(S,A) = Gain(S,A)/ IntI(S,A)

Gini Index

- Gini index is also type of criterion that helps us to calculate information gain. It measures the impurity of the node and is calculated for binary values only.

$$GI = 1 - \sum_{i=1}^{n} (p)^2$$

C1 = 0 , C2 = 6

P(C1) = 0/6 = 0

P(C2) = 6/6 = 1

$$Gini = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

- Gini impurity is more computationally efficient than entropy.

Implementation of K means clustering in Python

```
#First, start with importing necessary python packages

import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
from sklearn import tree


# read Iris dataset from csv file from your PC
iris=load_iris()
```

- To see all the features in the datset, use the print function

  print(iris.feature_names)

- To see all the target names in the dataset-

  print(iris.target_names)

- Now, we will remove the elements in the 0th, 50th, and 100th position. 0th element belongs to the Setosa species, 50th belongs Versicolor species and the 100th belongs to the Virginica species.
  - This will remove the labels for us to train our decision tree classifier better and check if it is able to classify the data well.
  - Spilitting the dataset

    ```
    removed =[0,50,100]
    new_target = np.delete(iris.target,removed)
    new_data = np.delete(iris.data,removed, axis=0)
    ```

- Train the Decision Tree Classifier, train classifier

```
clf = tree.DecisionTreeClassifier() # defining decision tree classifier
clf=clf.fit(new_data,new_target) # train data on new data and new target
prediction = clf.predict(iris.data[removed]) #  assign removed data as input
```

- Finally, we check if our predicted labels match the original labels

```
print("Original Labels",iris.target[removed])
print("Labels Predicted",prediction)
print("Accuracy = {}".format(accuracy*100))
```

```
Original Labels [0 1 2]
Labels Predicted [0 1 2]
Accuracy = 100.0
```