8. ID3 (Decision Tree) Algorithm

September 26, 2018

Saumay Agrawal
16BCE1151

In [1]:
```
from math import log
from pprint import pprint
import numpy as np
import pandas as pd
```

```
/anaconda3/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: numpy.dtype size changed
  return f(*args, **kwds)
/anaconda3/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: numpy.dtype size changed
  return f(*args, **kwds)
```

In [2]:
```
# Creating a Decision Tree class

class DecisionTree(object):

    # Constructor to initialize the dataframe of class
    def __init__(self, df):
        self.df = df
        self.entropy = self.getentropy(df.iloc[:, -1])

    def getdata(self):
        print(self.df)

    def proportion(self, c, n):
        if c == 0:
            return 0
        pc = c/n
        return -pc*log(pc, 2)

    # Compute the entroy of a given value vector
    def getentropy(self, values):
        n = len(values)
        value_dist = values.value_counts()
        entropy = 0
        for i in value_dist.index:
```

```
        entropy += self.proportion(value_dist[i], n)
    return entropy

# Compute the entropy for each attribute of dataset
def getgain(self, df):
    attribute = df.columns[0]
    info = {}
    avg = 0
    n = len(df.iloc[:, 0])
    value_dist = df.iloc[:, 0].value_counts()
    for i in value_dist.index:
        x = df.loc[df[attribute] == i]
        entropy = self.getentropy(x.iloc[:, -1])
        info[i] = entropy
        y = value_dist[i]/n * entropy
        avg += y
    info['average'] = avg
    return info

# Compute the info gain for each attribute of the dataset
def getinfogain(self):
    df = self.df
    infogain = {}
    target = df.columns[-1]
    for col in df.columns[:-1]:
        x = df.loc[:, [col, target]]
        info = self.getgain(x)
        infogain[col] = self.entropy - info['average']
    return infogain

# Find the root attribute having minimum infogain
def findroot(self):
    ifgain = self.getinfogain()
    attr, maxig = None, 0
    for k in ifgain.keys():
        if ifgain[k] > maxig:
            maxig = ifgain[k]
            attr = k
    return attr

# Create the decision tree recursively
def createTree(self):
    df = self.df
    if self.entropy == 0:
        target = df.iloc[:, -1].value_counts().index[0]
        return target
    rootattr = self.findroot()
    tree = {}
```

```
                for i in df[rootattr].value_counts().index:
                    cols = list(df.columns)
                    cols.remove(rootattr)
                    x = df.loc[df[rootattr]==i, cols]
                    subtree = DecisionTree(x)
                    tree[i] = subtree.createTree()
                return {rootattr: tree}
```

In [3]:
```
# Initialising the class with tennis data

data = pd.read_csv('data.csv')
dtree = DecisionTree(data)
dtree.getdata()
```

	outlook	temp	humidity	windy	play
0	sunny	hot	high	weak	no
1	sunny	hot	high	strong	no
2	overcast	hot	high	weak	yes
3	rainy	mild	high	weak	yes
4	rainy	cool	normal	weak	yes
5	rainy	cool	normal	strong	no
6	overcast	cool	normal	strong	yes
7	sunny	mild	high	weak	no
8	sunny	cool	normal	weak	yes
9	rainy	mild	normal	weak	yes
10	sunny	mild	normal	strong	yes
11	overcast	mild	high	strong	yes
12	overcast	hot	normal	weak	yes
13	rainy	mild	high	strong	no

In [4]:
```
# Display the computed Desicion Tree

pprint(dtree.createTree())
```

```
{'outlook': {'overcast': 'yes',
             'rainy': {'windy': {'strong': 'no', 'weak': 'yes'}},
             'sunny': {'humidity': {'high': 'no', 'normal': 'yes'}}}}
```