**CS 5402 Intro to DataMining**

**Cole Davis**

**Homework #2**

**Question #1:**

Table

Description automatically generated

**Question #2:** Graphical user interface, table

Description automatically generated

Diagram

Description automatically generated

**Question #3:**

Country\_likelyhood = 2/3 \* 1/3 \* 1/3 \* 3/8

Not\_Country\_likelyhood = (1/3 \* 2/3 \* 0/3 \* 3/8) \* (0/2 \* ½ \* ½ \* 2/8)

Convert to probabilities by normalizing so they sum to 1:

Final Answer Probability = Country\_likelyhood / (Country\_likelyhood + Not\_Country\_likelyhood)

**Question #4:**

Graphical user interface, application

Description automatically generated

Diagram

Description automatically generated

**Question #5a:**

entropyBeforeSplit = -3/8\*log(3/8) – ¼\*log(1/4) – 3/8\*log(3/8)

**Question #5b:**

entropyMystery = -1/4 \* log(1/4) – ¼ \* log(1/4) – ½ \* log(1/2)

**Question #5c:**

informationGain = X – (1/2 \* Y + ½ \* Z)

**Question #6a:**

**P(outlook = good) = 5/10**

P(outlook = good and play = yes) = 2/5

P(outlook = good and play = no) = 3/5

Gini index for outlook good = 1-((2/5)^2 + (3/5)^2) = **0.48**

**P(outlook = bad) = 5/10**

P(outlook = bad and play = yes) = 4/5

P(outlook = bad and play = no) = 1/5

Gini index for outlook bad = 1-((4/5)^2 + (1/5)^2) = **0.32**

**Weighted sum for outlook:** (5/10)\*0.48 + (5/10)\*0.32 = **0.4**

**P(temperature = warm) = 5/10**

P(temperature = warm and play = yes) = 2/5

P(temperature = warm and play = no) = 3/5

Gini index for temperature warm = 1-((2/5)^2 + (3/5)^2) = **0.48**

**P(temperature = cool) = 5/10**

P(temperature = cool and play = yes) = 4/5

P(temperature = cool and play = no) = 1/5

Gini index for temperature cool = 1-((4/5)^2 + (1/5)^2) = **0.32**

**Weighted sum for temperature:** (5/10)\*0.48 + (5/10)\*0.32 = **0.4**

**P(humidity = high) = 5/10**

P(humidity = high and play = yes) = 1/5

P(humidity = high and play = no) = 4/5

Gini index for humidity high = 1-((1/5)^2 + (4/5)^2) = **0.32**

**P(humidity = normal) = 5/10**

P(humidity = normal and play = yes) = 5/5

P(humidity = normal and play = no) = 0/5

Gini index for humidity normal = 1-((5/5)^2 + (0/5)^2) = **0.0**

**Weighted sum for humidity:** (5/10)\*0.32 + (5/10)\*0.0 = **0.16**

**P(windy = TRUE) = 5/10**

P(windy = TRUE and play = yes) = 3/5

P(windy = TRUE and play = no) = 2/5

Gini index for windy TRUE= 1-((3/5)^2 + (2/5)^2) = **0.48**

**P(windy = FALSE) = 5/10**

P(windy = FALSE and play = yes) = 3/5

P(windy = FALSE and play = no) = 2/5

Gini index for windy FALSE= 1-((3/5)^2 + (2/5)^2) = **0.48**

**Weighted sum for windy:** (5/10)\*0.48 + (5/10)\*0.48 = **0.48**

The root of the tree will be **humidity** because it had the lowest weighted sum of all the attributes.

**Question #6b:**

Diagram

Description automatically generated

from sklearn import tree

import pandas as pd

import numpy

import graphviz

df = pd.read\_csv('hw2\_prob6\_Copy.csv')

r,c = df.shape

#Replace nominal attributes with numeric for doing the CART algorithm.

df = df.replace({'outlook': r'good'}, {'outlook':1}, regex=True)

df = df.replace({'outlook': r'bad'}, {'outlook':0}, regex=True)

df = df.replace({'temperature': r'warm'}, {'temperature':1}, regex=True)

df = df.replace({'temperature': r'cool'}, {'temperature':0}, regex=True)

df = df.replace({'humidity': r'high'}, {'humidity':1}, regex=True)

df = df.replace({'humidity': r'normal'}, {'humidity':0}, regex=True)

df = df.replace({'windy': r'TRUE'}, {'windy':1}, regex=True)

df = df.replace({'windy': r'FALSE'}, {'windy':0}, regex=True)

X = df.iloc[:, 0:c-1].values    # non-decision attributes

y = df.iloc[:, c-1].values      # decision attribute

clf = tree.DecisionTreeClassifier(criterion="gini")

clf = clf.fit(X,y)

attrNames = list(df.columns)

classNames = list(set(df["play"].values))

classNames.sort()

classNames = numpy.array(classNames)

dot\_data = tree.export\_graphviz(

    clf,

    out\_file=None,

    feature\_names=attrNames[0:c-1],

    class\_names=classNames,filled=True,

    rounded=True,

    special\_characters=True)

graph = graphviz.Source(dot\_data)

graph.render("Trading\_Decision\_Tree") # see Trading\_Decision\_Tree.pdf

**Question #6c:**

Graphical user interface, text

Description automatically generated