Name: Solution Key **30 points possible**

CS 5402 – Intro to Data Mining
Fall 2020
HW #2

Submit as a single pdf file via Canvas by 11:59 p.m. on Sep. 21, 2020

1. Consider the following dataset where the decision attribute is ***restaurant***:

mealPreference	gender	drinkPreference	restaurant
hamburger	M	coke	mcdonalds
fish	M	pepsi	burgerKing
chicken	F	coke	mcdonalds
hamburger	M	coke	mcdonalds
chicken	M	pepsi	wendys
fish	F	coke	burgerKing
chicken	M	pepsi	burgerKing
chicken	F	coke	wendys
hamburger	F	coke	mcdonalds

Use the **1-rule (1R) method** to find the best single attribute to determine ***restaurant***. In order to demonstrate that you actually know how this method works (and aren't just guessing at which attribute is best), you **must** fill in **ALL** of the blank values in the table below; otherwise, you will **not** receive any credit for this problem. **(10 pts.)**

Attribute	Attribute Value	# Rows with Attribute Value	Most Frequent Value for *restaurant*	Errors	Total Errors
mealPreference	hamburger	3	**mcdonalds (3)**	**0**	**2**
	fish	2	**burgerKing (2)**	**0**	
	chicken	4	**wendys (2)**	**2**	
gender	M	5	**mcdonalds or burgerKing (2)**	**3**	**5**
	F	4	**mcdonalds (2)**	**2**	
drinkPreference	pepsi	3	**burgerKing (2)**	**1**	**3**
	coke	6	**mcdonalds (4)**	**2**	

Based on **these** calculations, list the **rules** that would be generated by the **1R** method for determining ***restaurant***.

Name: <u>Solution Key</u> **30 points possible**

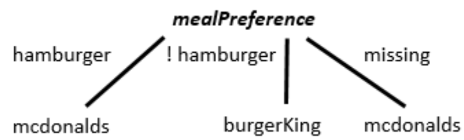
Fewest errors are for *mealPreference*, so rules would be:

If *mealPreference* = hamburger then *restaurant* = mcdonalds
If *mealPreference* = fish then *restaurant* = burgerKing
If *mealPreference* = chicken then *restaurant* = wendys

Each entry in the table is worth ½ pt. and each rule is worth ½ pt.

2. Create the dataset given in problem 1. as an *arff* or *csv* file, and run **DecisionStump** on it in **Weka**. List the classification rules that are produced (you can just include a screenshot of your Weka output) <u>**AND**</u> draw a tree that corresponds to the rules. **(1.5 pts.)**

```
Decision Stump

Classifications

mealPreference = hamburger : mcdonalds
mealPreference != hamburger : burgerKing
mealPreference is missing : mcdonalds
```



Weka rules worth ½ pt. and hand-drawn tree worth 1 pt.

3. **Statistical modeling** can be used to compute the probability of occurrence of an attribute value. Based on the data given in the table below, if we have a new instance where ***ageGroup* = youngAdult**, ***gender* = M**, and ***bookPreference* = mystery**, what is the <u>**likelihood**</u> that ***musicPreference* = rock**? Just <u>**set up the equation**</u> to compute this; don't actually evaluate the equation. **(1 pt.)**

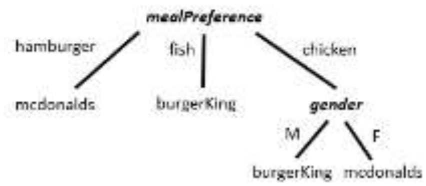
ageGroup				gender				bookPreference				musicPreference		
	rock	classical	country		rock	classical	country		rock	classical	country	rock	classical	country
youngAdult	1	0	2	M	2	1	1	sciFiction	2	0	0	3	2	3
middleAge	2	0	1	F	1	1	2	mystery	1	1	2			
senior	0	2	0					nonFiction	0	1	1			
youngAdult	1 / 3	0 / 2	2 / 3	M	2 / 3	1 / 2	1 / 3	sciFiction	2 / 3	0 / 2	0 / 3	3 / 8	2 / 8	3 / 8
middleAge	2 / 3	0 / 2	1 / 3	F	1 / 3	1 / 2	2 / 3	mystery	1 / 3	1 / 2	2 / 3			
senior	0 / 3	2 / 2	0 / 3					nonFiction	0 / 3	1 / 2	1 / 3			

1/3 * 2/3 * 1/3 * 3/8 (1/4 pt. per term)

4. Create the dataset given in problem 1. as an *arff* or *csv* file, and run **Id3** on it in **Weka**. Show the decision tree output that is produced by Weka **AND** draw the tree by hand. **(1.5 pts.)**

   Note: Id3 may not be installed with the initial download of Weka 3.8, in which case you will need to install the package named *simpleEducationalLearningSchemes*.

```
Id3

mealPreference = hamburger: mcdonalds
mealPreference = fish: burgerKing
mealPreference = chicken
|  gender = M: burgerKing
|  gender = F: mcdonalds
```



Weka output worth ½ pt. and tree worth 1 pt.

Name: <u>Solution Key</u> **30 points possible**

5. Consider the following dataset where the decision attribute is ***restaurant***:

mealPreference	gender	drinkPreference	restaurant
hamburger	M	coke	mcdonalds
fish	M	pepsi	burgerKing
chicken	F	coke	mcdonalds
hamburger	M	coke	mcdonalds
chicken	M	pepsi	wendys
fish	F	coke	burgerKing
chicken	M	pepsi	burgerKing
chicken	F	coke	wendys
fish	F	coke	mcdonalds
hamburger	F	coke	mcdonalds

If we want to make an **ID3 decision tree** for determining ***restaurant***, we must decide which of the three non-decision attributes (***mealPreference, gender,*** or ***drinkPreference***) to use as the root of the tree.

a. Set up the **equation** to compute what in lecture we called **entropyBeforeSplit** for ***restaurant***. You do **<u>not</u>** have to actually solve (i.e., evaluate the terms in) the equation, just set up the equation with the appropriate values. **(2 pts.)**

p(mcdonalds) = 5/10 p(burgerKing) = 3/10 p(wendys) = 2/10

entropyBeforeSplit = -5/10*log(5/10) - 3/10*log(3/10) - 2/10*log(2/10)

b. Set up the **equation** to compute **entropy** for ***mealPreference*** when its value is **chicken**. That is, a tree with ***mealPreference*** at the root would have three branches (one for **hamburger**, one for **chicken,** and one for **fish**), requiring us to compute **entropyHamburger, entropyChicken,** and **entropyFish**; here we only want you to set up the equation to compute **entropyChicken**. You do **<u>not</u>** have to actually solve (i.e., evaluate the terms in) the equation, just set it up using the appropriate values. **(2 pts.)**

chicken [1 mcdonalds, 2 wendys, 1 burgerKing]

p(mcdonalds) = 1/4 p(wendys) = 2/4 p(burgerKing) = 1/4

entropyChicken = -1/4*log(1/4) - 2/4*log(2/4) - 1/4*log(1/4)

c. Suppose that instead of considering ***mealPreference*** to be the root of this decision tree, we had instead considered ***drinkPreference***. Set up the **equation** to compute **information gain** for ***drinkPreference*** given the **<u>variables</u> (X, P, and C)** specified below. **(2 pts.)**

entropy before any split: **X**
entropy for ***drinkPreference* = pepsi:** **P**
entropy for ***drinkPreference* = coke:** **C**
entropyAfterSplit = (3/10 * P) + (7/10 * C)
infoGain = X – entropyAfterSplit

Name: Solution Key **30 points possible**

6. Consider the following dataset where the decision attribute is ***isSticky***:

consistency	packaging	chocolate	isSticky
soft	individuallyWrapped	yes	yes
soft	box	yes	no
hard	individuallyWrapped	no	yes
hard	box	no	yes
hard	box	yes	no
soft	individuallyWrapped	no	yes

Test subject for this study

a. Do **only** the necessary calculations to determine what **the root node** would be for a **CART decision tree**. **YOU MUST SHOW YOUR WORK!!! (5 pts.)**

Note: If there's a tie for which attribute you'd pick to be the root of the tree, just list those attributes and say that we could pick from them.

P(consistency = soft) = 3/6
P(consistency = soft and isSticky = no) = 1/3
P(consistency = soft and isSticky = yes) = 2/3
Gini index for consistency = soft: 1- $((1/3)^2 + (2/3)^2)$ = 0.45
P(consistency = hard) = 3/6
P(consistency = hard and isSticky = no) = 1/3
P(consistency = hard and isSticky = yes) = 2/3
Gini index for consistency = hard: 1- $((1/3)^2 + (2/3)^2)$ = 0.45
Weighted sum for consistency: (3/6)*0.45 + (3/6)*0.45 = 0.45 (1.5 pts.)

P(packaging = individuallyWrapped) = 3/6
P(packaging = individuallyWrapped and isSticky = no) = 0/3
P(packaging = individuallyWrapped and isSticky = yes) = 3/3
Gini index for packaging = individuallyWrapped: 1- $((0/3)^2 + (3/3)^2)$ = 0
P(packaging = box) = 3/6
P(packaging = box and isSticky = no) = 2/3
P(packaging = box and isSticky = yes) = 1/3
Gini index for packaging = box: 1- $((2/3)^2 + (1/3)^2)$ = 0.45
Weighted sum for packaging: (3/6)*0 + (3/6)*0.45 = 0.225 (1.5 pts.)

P(chocolate = yes) = 3/6
P(chocolate = yes and isSticky = no) = 2/3
P(chocolate = yes and isSticky = yes) = 1/3
Gini index for chocolate = yes: 1- $((2/3)^2 + (1/3)^2)$ = 0.45
P(chocolate = no) = 3/6
P(chocolate = no and isSticky = no) = 0/3
P(chocolate = no and isSticky = yes) = 3/3
Gini index for chocolate = no: 1- $((0/3)^2 + (3/3)^2)$ = 0

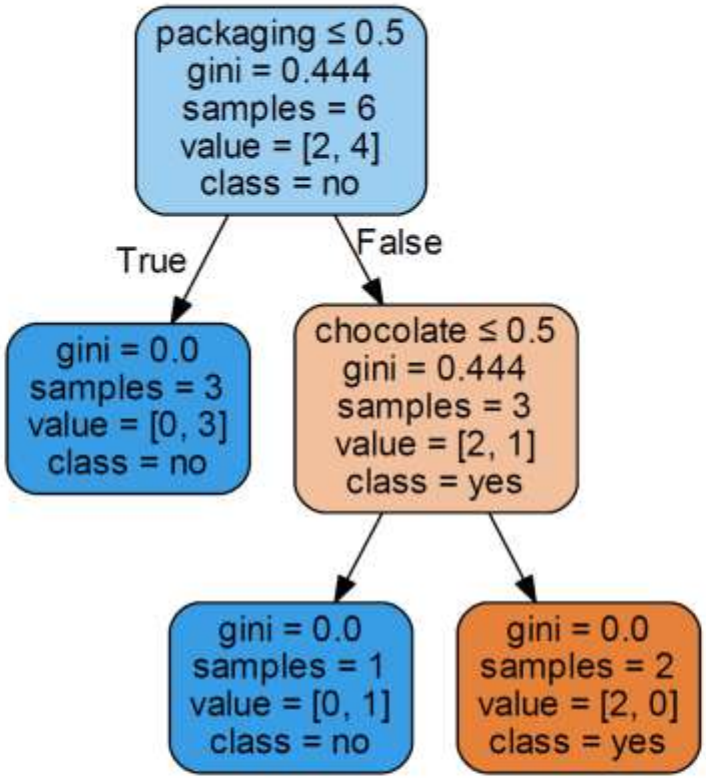
Weighted sum for packaging: (3/6)*0.45 + (3/6)*0 = 0.225 (1.5 pts.)

Lowest weighted sum tied between packaging and chocolate; either could be the root of the tree (0.5 pt.)

b. Write a **Python** program that runs the **CART algorithm** on this dataset. Include both your **source code and a screenshot** showing the resulting tree. The dataset (candy.csv) is posted on Canvas along with this assignment. **(4 pts.)**

```
from sklearn import tree
import pandas as pd
import numpy
import graphviz

df = pd.read_csv('candy.csv')
r,c = df.shape

# Non-decision attr's have to be numeric!
# Doesn't matter what you assign as values
df= df.replace({'consistency': r'soft'}, {'consistency':0}, regex=True)
df= df.replace({'consistency': r'hard'}, {'consistency':1}, regex=True)
df= df.replace({'packaging': r'individuallyWrapped'}, {'packaging':0},
regex=True)
df= df.replace({'packaging': r'box'}, {'packaging':1}, regex=True)
df= df.replace({'chocolate': r'yes'}, {'chocolate':1}, regex=True)
df= df.replace({'chocolate': r'no'}, {'chocolate':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 = numpy.array(list(set(df["isSticky"].values)))

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("Candy_Decision_Tree")
# see Candy_Decision_Tree.pdf
```

(3 pts.)



(1 pt.)

c. Run **SimpleCart** in **Weka** on this dataset specifying the option **usePrune = False**. Show a screenshot of the **CART decision tree** that it produces. **(1 pt.)**

Note: SimpleCART may not be installed with the initial download of Weka 3.8, in which case you will need to install the package.

```
CART Decision Tree

packaging=(box): no(2.0/1.0)
packaging!=(box): yes(3.0/0.0)
```