CS 5402 Intro to DataMining

Cole Davis

Homework #2

<u>Question #1:</u>

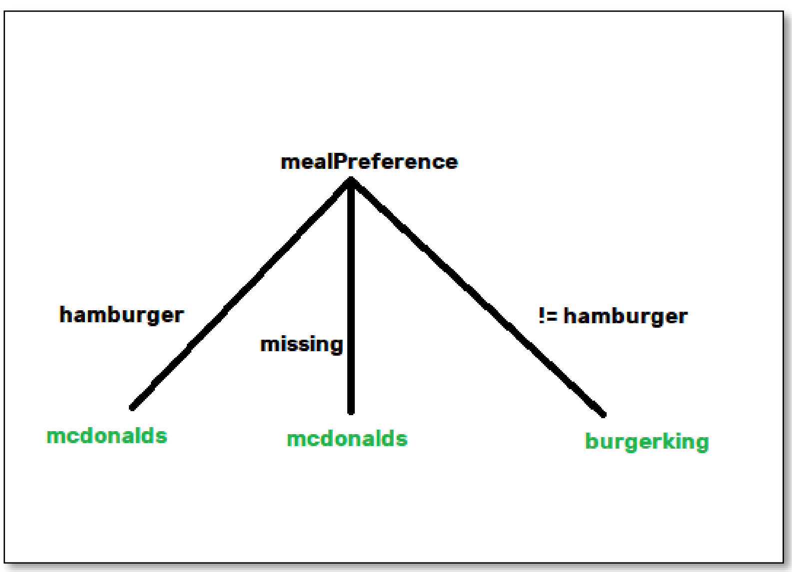
Attribute	Attribute Value	# Rows with Attribute Value	Most Frequent Value for *restaurant*	Errors	Total Errors
mealPreference	hamburger	3	mcdonalds(3)	0	2
	fish	2	burgerking(2)	0	
	chicken	4	wendys(2)	2	
gender	M	5	mcdonalds(2)	3	5
	F	4	mcdonalds(2)	2	
drinkPreference	pepsi	3	burgerking(2)	1	3
	coke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

Rules Generated:

mealPreference = hamburger -> mcdonalds
mealPreference = fish -> burgerking
mealPreference = chicken -> wendys

Question #2:

```
Class distributions

ï»¿mealPreference = hamburger
mcdonalds	burgerking	wendys
1.0	0.0	0.0
ï»¿mealPreference != hamburger
mcdonalds	burgerking	wendys
0.16666666666666666	0.5	0.3333333333333333
ï»¿mealPreference is missing
mcdonalds	burgerking	wendys
0.4444444444444444	0.3333333333333333	0.2222222222222222

Time taken to build model: 0 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances           6               66.6667 %
Incorrectly Classified Instances         3               33.3333 %
Kappa statistic                          0.4706
Mean absolute error                      0.2716
Root mean squared error                  0.3685
Relative absolute error                 62.8571 %
Root relative squared error             79.5672 %
Total Number of Instances                9

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.750    0.000    1.000      0.750    0.857      0.791    0.875     0.861     mcdonalds
                 1.000    0.500    0.500      1.000    0.667      0.500    0.750     0.500     burgerking
                 0.000    0.000    ?          0.000    ?          ?        0.714     0.333     wendys
Weighted Avg.    0.667    0.167    ?          0.667    ?          ?        0.798     0.623

=== Confusion Matrix ===

 a b c   <-- classified as
 3 1 0 | a = mcdonalds
 0 3 0 | b = burgerking
 0 2 0 | c = wendys
```



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)

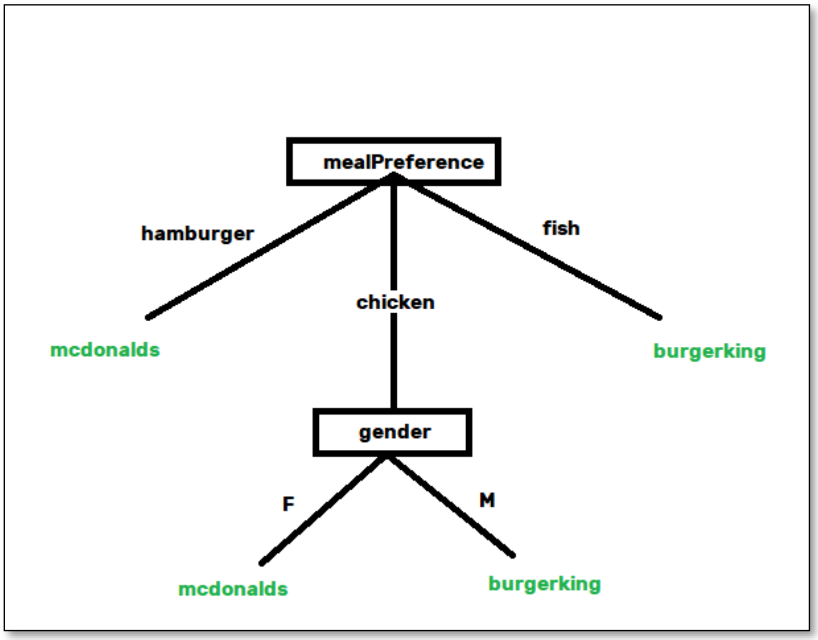
Convert to probabilities by normalizing so they sum to 1:

Final Answer Probability = Country_likelyhood / (Country_likelyhood + Not_Country_likelyhood)

Question #4:

Classifier output

```
                 restaurant
Test mode:       evaluate on training data

=== Classifier model (full training set) ===

Id3

ï»¿mealPreference = hamburger: mcdonalds
ï»¿mealPreference = fish: burgerking
ï»¿mealPreference = chicken
|   gender = M: burgerking
|   gender = F: mcdonalds

Time taken to build model: 0 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances           7               77.7778 %
Incorrectly Classified Instances         2               22.2222 %
Kappa statistic                          0.6327
Mean absolute error                      0.1481
Root mean squared error                  0.2722
Relative absolute error                 34.2857 %
Root relative squared error             58.7643 %
Total Number of Instances                9

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.200    0.800      1.000    0.889      0.800    0.975     0.950     mcdonalds
                 1.000    0.167    0.750      1.000    0.857      0.791    0.972     0.917     burgerking
                 0.000    0.000    ?          0.000    ?          ?        0.857     0.500     wendys
Weighted Avg.    0.778    0.144    ?          0.778    ?          ?        0.948     0.839

=== Confusion Matrix ===

 a b c   <-- classified as
 4 0 0 | a = mcdonalds
 0 3 0 | b = burgerking
 1 1 0 | c = wendys
```



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)

<u>Question #6a:</u>

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

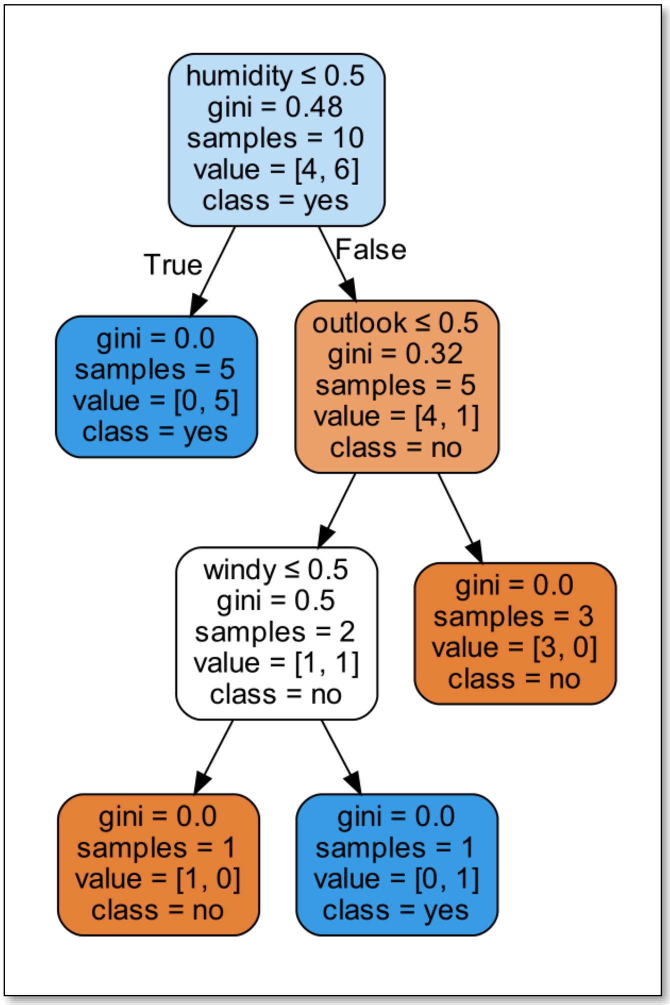
P(windy = FALSE and play = no) = 2/5

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:




```
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)
```

```python
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:

Classifier output

```
=== Classifier model (full training set) ===

CART Decision Tree

humidity=(normal): yes(5.0/0.0)
humidity!=(normal)
|   outlook=(bad)
|   |   temperature=(cool): yes(1.0/0.0)
|   |   temperature!=(cool): no(1.0/0.0)
|   outlook!=(bad): no(3.0/0.0)

Number of Leaf Nodes: 4

Size of the Tree: 7

Time taken to build model: 0 seconds
```