Decision Trees & Entropy

John Smutny

Homework 3

ECE5984 Applications of ML

03/01/2022

Table of Contents

Abstract

This assignment walks through two separate use cases for calculating Decision Trees and seeing how the tree's criterion impacts the resulting Decision Tree.

The first section compares Decision Trees created using Information Gain versus Entropy to classify what types of found wild mushrooms are edible. In addition, the section walks through the mathematical process of creating a Decision Tree; calculating dataset Information Gain, choosing a root node, re-calculating next node Information Gain, etc.

The second section exclusively uses python programs and the SciKit-Learn module to compare the classifications of witnessed flares based on several characteristics. This section focuses on the difference in a resulting Decision Tree if the tree's criterion is Entropy versus Gini Index.

Definitions

- Information Gain: The magnitude of the reduction in Entropy when a feature is used to partition (split) a dataset.
    - Higher numbers mean that picking that feature will reduce the dataset the most. IE: Make the most impact to create a pure partition or (for large partitions) splitting the dataset closer to 50:50.
- Entropy: The amount of disorder/diversity in a dataset.
    - Higher numbers means that the dataset has more varied values and more possible outcomes.
- Gini Index: An alternative measure of impurity (compared to entropy) that is based on how likely a model will miss classify data.

Creating Decision Trees

Decision Tree 1: Hand Calculations

Instructions: *“Using the Information Gain-based method described in class, determine the best decision tree to embody the following information. White, Tall and Frilly are the attributes and Edible is the result or output that your decision tree must generate.*
You should use the criterion of maximum information gain to select the attributes to use in each level of the tree, starting from the top. You must show all of your work. Draw the resulting tree.”

See Appendix for the exact hand calculations. Please see below for a summary of information at each step.

Dataset Summary

Feature Variable(s): { White (W), Tall (T), Frilly (F) } (all boolean)
Target Variable(s): { Edible } (boolean)

Decisions (I): { Edible (E) , Not Edible (N) }

Number of Samples: 24
- { Edible (E) , Not Edible (N) } = { 16, 8 }
- { White (W), Tall (T), Frilly (F) } = { 10, 14, 8 }

Node Selection

Root Node

H(t, D) = 0.918 bits
rem(W, D) = 0.367
rem(T, D) = 0.503
rem(F, D) = 0.262

IG_{White} = 0.551 bits
IG_{Tall} = 0.415 bits
IG_{Frilly} = 0.656 bits

Conclusion:
Boolean feature ‘Frilly’ is selected as the root node since the feature provides the greatest amount of Information Gain. IE: Reduces the Entropy (disorder) of the system the most.

Second Node

A) Frilly = TRUE

$H(t, d_{Frilly=TRUE}) = 0.954$ bits

rem(W, D) = 0.000

rem(T, D) = 0.451

IG_{White} = 0.954 bits

IG_{Tall} = 0.503 bits

Conclusion A:
Boolean feature ‘Frilly’ is selected as the root node

B) Frilly = FALSE

$H(t, d_{Frilly=FALSE}) = 0.696$ bits

rem(W, D) = 0.000

rem(T, D) = 0.000

IG_{White} = 0.696 bits

IG_{Tall} = 0.696 bits

Conclusion B:
Either boolean feature (White or Tall) can be used as a secondary node. Feature ‘White’ was selected.

Third Node

C) Frilly = TRUE, White = FALSE

$H(t, d_{Frilly=TRUE, White=FALSE}) = 0.971$ bits

rem(T, D) = 0.551

IG_{Tall} = 0.420 bits

Conclusion C:
Boolean feature ‘Tall’ is the last remaining feature to partition the dataset. There is still remaining Entropy which means that the resulting data partition at this node is not pure. This is not inherently an issue, but something to understand when analyzing the resulting tree on the next page.

D) Frilly = FALSE, White = FALSE

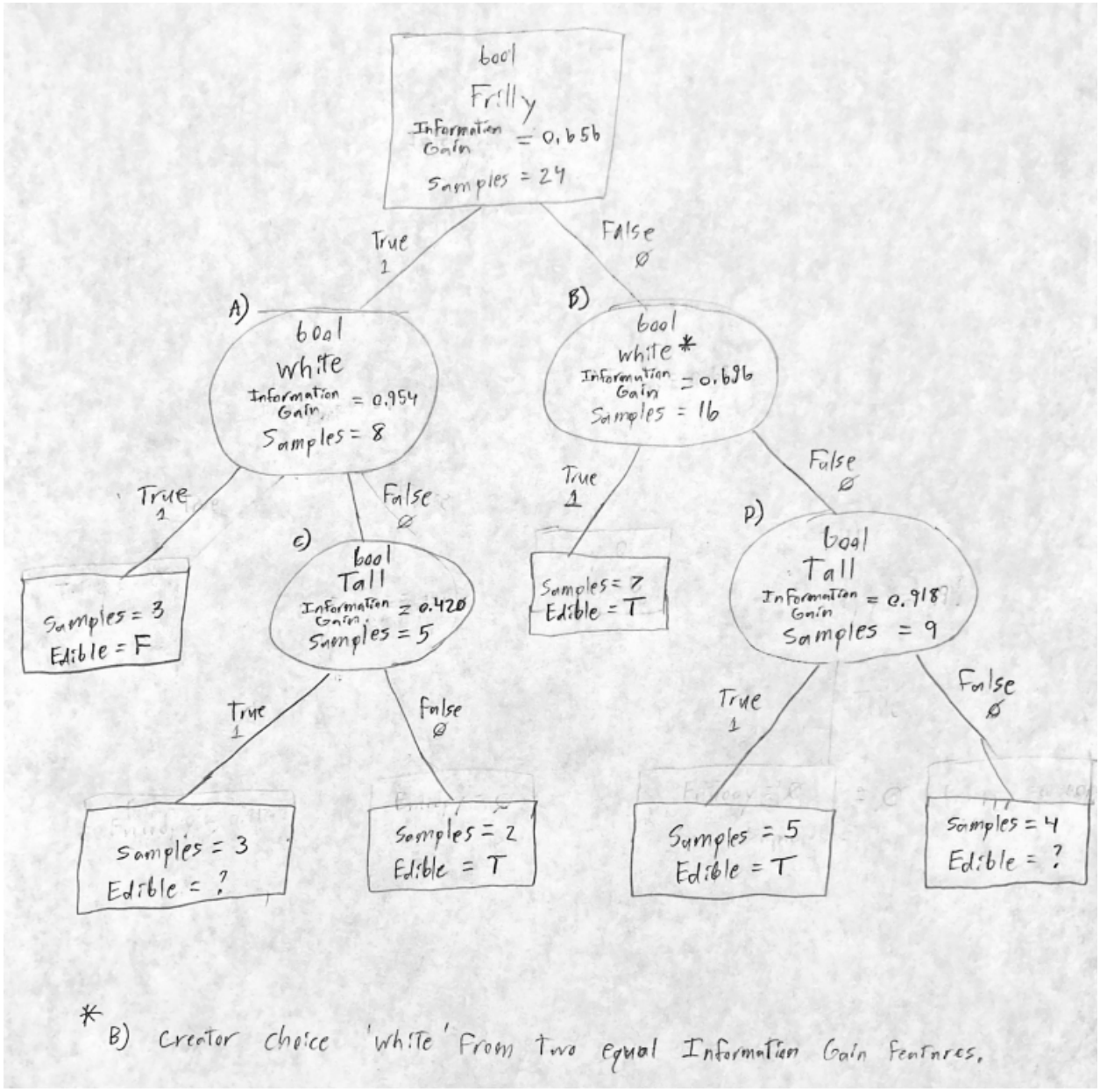
$H(t, d_{Frilly=FALSE, White=FALSE}) = 0.918$ bits

rem(T, D) = 0.000

IG_{Tall} = 0.918 bits

Conclusion D:
Boolean feature ‘Tall’ is the last remaining feature to partition the dataset, however the remaining entropy for the ‘Tall = TRUE’ partition leads to a pure set. The remaining ‘Tall = FALSE’ partition does not lead to a pure set.

Information Gain Tree



Decision Tree 2: Python & Scikit-Learn

Instructions: *In the Datasets folder in the Files area of our Canvas site, you will find an Excel spreadsheet called "AlienMushrooms.xlsx". This contains the dataset represented above. You are to write a simple Python program to read this spreadsheet and to generate a scikit-learn Decision Tree classifier for the data. Use the "entropy" criterion for the DT generation. Use the "writegraphtofile" function I have provided below to display the resulting tree, and compare it to your manual results above.*

Please see the Appendix "Python Program: Mushroom Decision Tree Classifier" for the python code used to generate Figure2

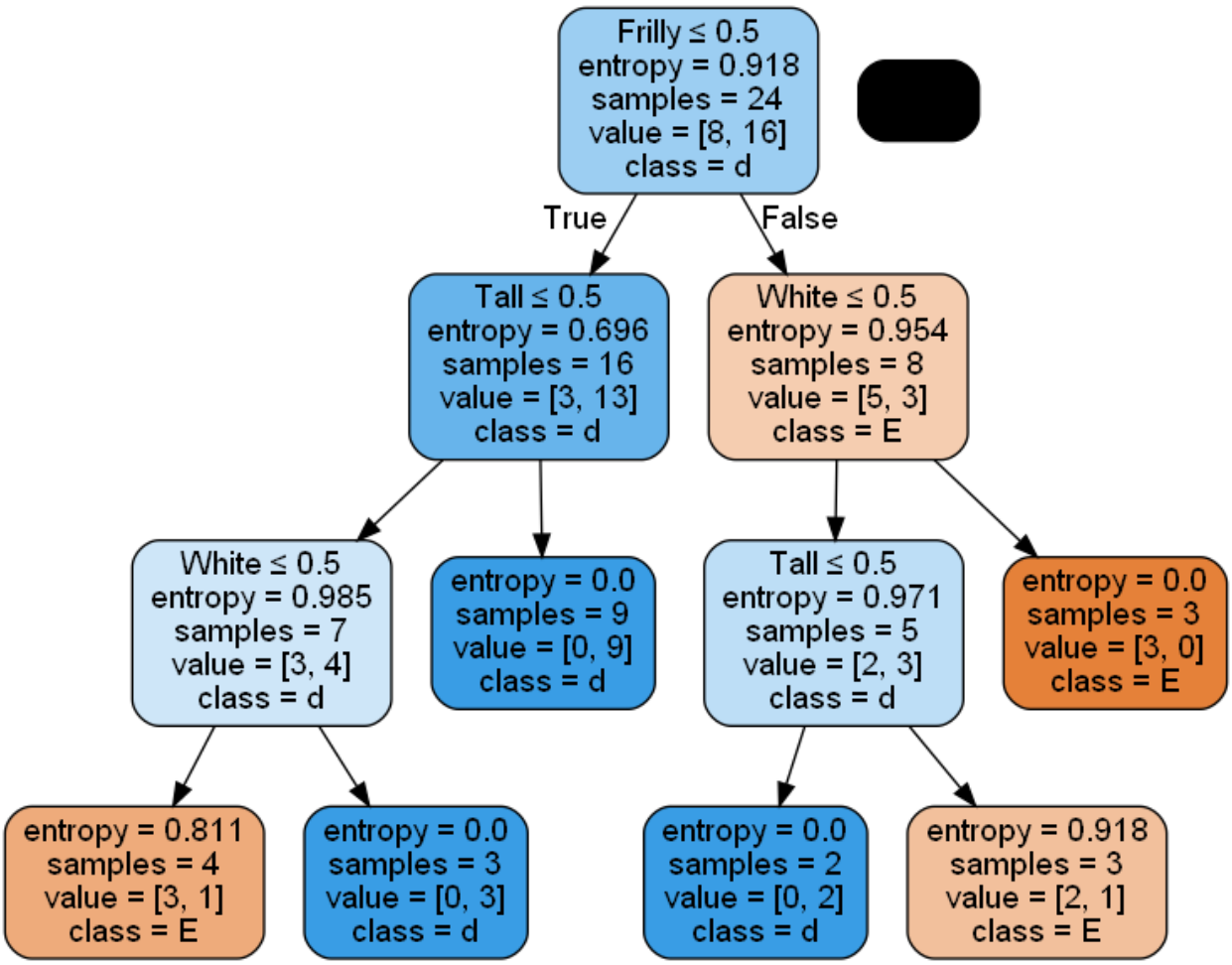


Figure 2: Decision Tree of edible mushrooms based on the dataset feature's Entropy.

Discussion

Topic: Discussion of differences between your parts 1 and 2

The trees created from using the ‘Information Gain’ versus ‘Entropy’ criterion are similar in depth and in end result, but different in the features used in the second level nodes (se annotation * below).

Both trees use all three boolean features and both trees lead to four pure sets and two sets with remaining uncertainty. This is encouraging that both methods lead to similar models, since both models are very closely related. ‘Information Gain’ is a part of determining the remaining ‘Entropy’ of a system and that one cannot be used without the other.

These trees differ primarily in what features were used in the second level of the tree. The ‘Entropy’ and ‘Information Gain’ trees potentially are opposite of one another (see * below); when Frilly=True, the ‘Information Gain’ model uses the ‘White’ feature’, but the ‘Entropy’ model uses the ‘Tall’ feature.

*Side observation; During hand calculations on the second level, after the higher Information Gain feature (‘Frilly’) was chosen, the remaining dataset partition had a greater dataset Entropy value than at the start of the process. The python decision model confirms this behavior.

Decision Tree 3: Entropy vs Gini

Instructions: *"In the Datasets folder in the Files area of our Canvas site, you will find an Excel spreadsheet called "FlareData.xlsx". This contains the dataset for this homework assignment. You are to do the following:*

1. *The predictors are all categorical; to use the decision tree capabilities of scikit-learn, you will need to replace the letter or text labels with numeric values. You may do this manually in the spreadsheet OR in your Python code – either approach is just fine.*
    a. *(note: in general, we need to be VERY careful in doing this, and we will discuss this again. Since we are using a decision tree model, this approach is less risky).*

2. *In this dataset there are three possible target values that we might want to predict: C class, M class and X class. For this assignment let's concentrate on the C class as our target variable. (Note: M and X cannot be used as predictors).*

Data Exploration

In order to model the desired features and target from the given data; several steps were taken to prepare the data for modeling.

A) From visual inspection; there are zero instances of invalid data, so no action is required to ensure correct data.

B) Target variable columns { M Class, X Class } were deleted from the dataset. Only the 'C Class' target variable will be used in this modeling activity to compare Entropy and Gini Index decision tree constructions.

C) Convert the data used to only a numeric format. Data in feature columns { Zurich Class, Spot Size, and Spot Distance } all are categorical character values. Each character value was replaced with the following numeric value as defined below in the dataset.

Original Categorical Value	→	**Numeric Value**	**Number of Instances Effected**
Zurich Class			323
B		0	65
C		1	76
D		2	88
E		3	21
F		4	8
H		5	65

Original Categorical Value	→	Numeric Value	Number of Instances Effected
Spot Size			323
A		0	61
H		1	11
K		2	48
R		3	38
S		4	100
X		5	65
Spot Distance			323
C		0	19
I		1	89
O		2	150
X		3	65

Comparing Decision Trees (Entropy vs Gini)

3. *Use python to develop code to read the data tab from the spreadsheet, create two decision tree classifiers and write the resulting trees to a png file. One tree should use the entropy criterion, while the other should use the gini criterion. Limit the depth of the trees to four. Below you will see a simple Python function to write a decision tree classifier to a png image file. Compare the two trees: which one is better? Why?"*

Please see the Appendix "Python Program: * Flare Tree Classifier" for the python code used to generate both Figure3A and Figure3B.

Entropy Decision Tree

- Criterion: Entropy
- Max Depth: 4

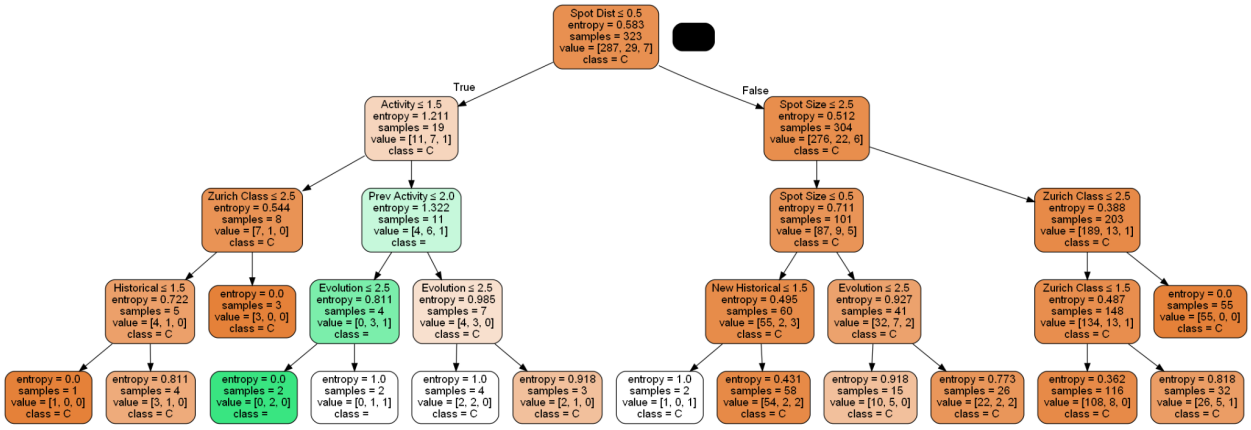


Figure 3A: FlareData Decision Tree created based on the dataset feature's Entropy

Gini Index Decision Tree

- Criterion: Gini Index
- Max Depth: 4

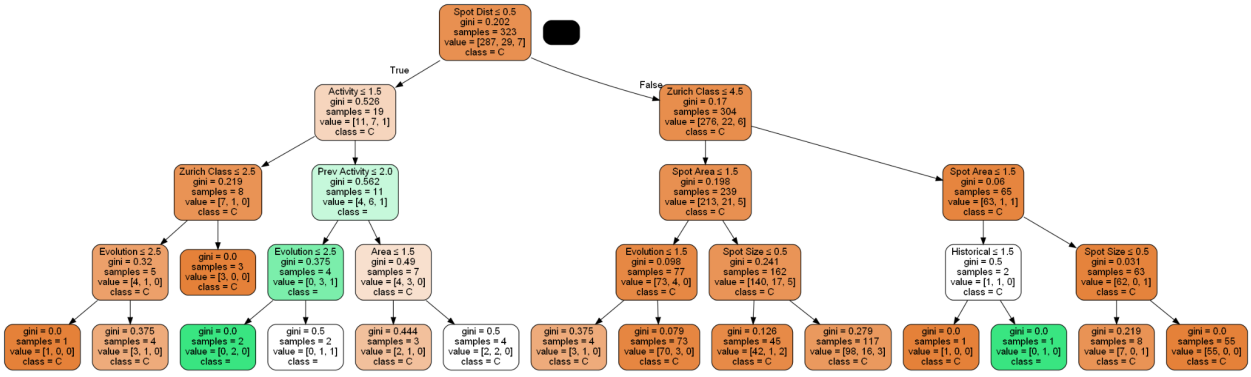


Figure 3B: FlareData Decision Tree created based on the dataset feature's Gini Index

Discussion

Topic: Discussion of the differences between the two trees generated in part 3.

After creating two different Decision Trees based on different criterions (Entropy vs Gini Index); there are several observations about the tree's similarities and differences.

The trees are similar in A) overall shape (despite the depth restriction), B) they both chose the ordinal feature 'Spot Distance' as their root nodes (most impactful feature), C) had one particular identical partition. To start, the tree depth restriction (4) affected both trees equally, since there were a similar amount of leaf nodes. This implies that both methods of partitioning the data wouldn't have a significant impact on simplifying the dataset. Second, both methodologies used the ordinal 'Spot Distance' feature as the top feature when splitting the dataset into two subsets; both split into datasets with the 'Class C' distance and the 'Class I/O/X' distances. Lastly, on the left side, 2nd level, of both trees, there was a similar partition path from 'Activity' to 'Prev. Activity' to 'Evolution' to separate the subsets.

The major differences were most readily visible when you observed when 'Zunch Class' was used to partition the datasets. In the 2nd and 3rd levels the 'Zunch Class' criterion was used at different times depending on the methodology used. Please see Figure 3C for more details.

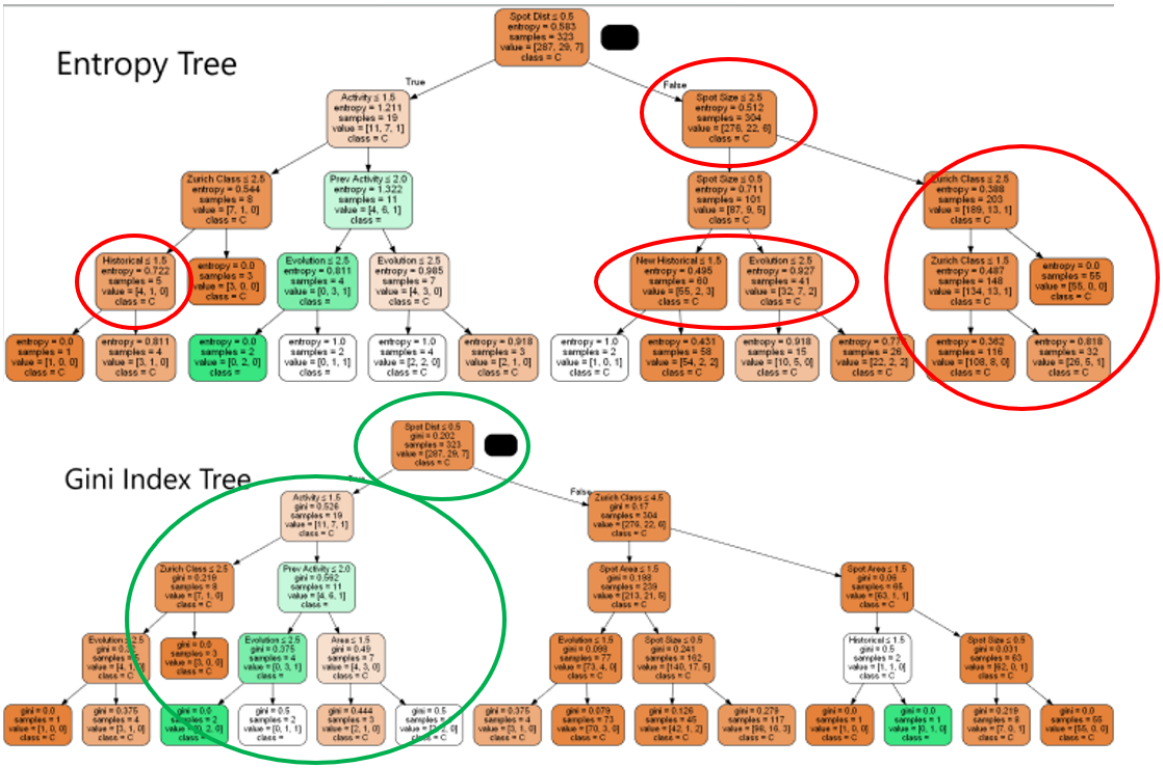


Figure 3C: Combined image pointing out some example similarities (green) and differences (red) between the Entropy and Gini Index based Decision Trees.

Appendix

Data set References

1. AlienMushrooms.xlsx
    a. Abstract: Boolean features of mushrooms that were eaten and a boolean result of if the mushroom was edible or not.
    b. Provider: Dr Creed Jones, Virginia Tech

2. FlareData.xlsx
    a. Abstract: Boolean and Categorical features that classify the type of flare witnessed.
    b. Provider: Dr Creed Jones, Virginia Tech

Python Program: Mushroom Decision Tree Classifier

```
#########################################################################
#   File:  hw_DecisionTrees_scikit_learn.py
#   Name:   John Smutny
#   Course: ECE-5984: Applications of Machine Learning
#   Date:   03/01/2022
#   Description:
#       Use the SciKit-Learn and imaging modules (pydot&graphviz) to create
#       a Decision Tree classifer for if mushrooms are edible based on
#       three features { Frilly, Tall, White } and their entropy on the
#       dataset.
#
#   Potential library fcts
#       1) fct to extract 'feature' labels and 'target' labels based on
#       inputted numbers (assuming that all features are on the left and all
#       targets are on the right).
#       2) Not hard code other items.
#########################################################################

# Decision tree modules
import pandas as pd             # Use Data Frames to organize read in data
from sklearn import tree        # Decision Tree functionality
import pydot                    # To print a resulting tree to a pdf image.
import graphviz


###################################
# Initial loading of data
filename = 'C:/Data/AlienMushrooms.xlsx'
df = pd.read_excel(filename, sheet_name='Data')

###################################
# Organize data to be trained. Separate Features and Target variables.
#   'Edible' is the last column and the dataset's Target variable.
x = df.drop('Edible', axis=1)   # Isolate the data features to train model.
y = df.get('Edible')                    # Isolate the target variable

featureLabels = x.columns.values
targetLabel = "Edible"

###################################
# Setup and train the classifier tree based on the data's entropy
calculations.
clf_Tree = tree.DecisionTreeClassifier(criterion='entropy')
clf_Tree = clf_Tree.fit(x, y)   # Train the model based on features and target

dot_data = tree.export_graphviz(clf_Tree,
```

```
                                out_file=None,
                                feature_names=featureLabels,
                                class_names=targetLabel,
                                filled=True,
                                rounded=True,
                                special_characters=True)

# Generate Results
graph = graphviz.Source(dot_data)
graph.render("Edible Mushrooms Tree")       #.pdf
(graph,) = pydot.graph_from_dot_data(dot_data)
graph.write_png("EdibleMushrooms_Tree.png") #.png
```

Python Program: Flare Decision Tree Classifier (Entropy & GI)

```
#########################################################################
#   File:   hw_DecisionTrees_Entropy_v_Gini.py
#   Name:   John Smutny
#   Course: ECE-5984: Applications of Machine Learning
#   Date:   03/01/2022
#   Description:
#       Use the SciKit-Learn module to explore how different data metrics can
#       influence a created Decision Tree; Specifically Entropy vs Gini Index.
#       This difference is explored with provided categorical data on solar
#       flares.
#       The target feature is the 'classification' of the flare based on
#       'One-Hot Binary Encoding'. { Class C, Class M, Class X }
#       For this program, ONLY CLASS C will be considered a Target (ignore
#       Class M & X)
#
#   Definitions:
#       Entropy: The amount of disorder/diversity in a dataset. Higher number
#                   means that the dataset has more varied values and more
#                   possible outcomes.
#       Gini Index: An alternative measure of impurity (compared to entropy)
#                   that is based on how likely a model will mis-classify
data.
#########################################################################

# Decision tree modules
import pandas as pd             # Use Data Frames to organize read in data
from sklearn import tree        # Decision Tree functionality
import pydot                    # To print a resulting tree to a pdf image.

###################################
# Initial loading of data
filename = 'C:/Data/FlareData_EDIT.xlsx'
df = pd.read_excel(filename, sheet_name='data')

###################################
# Data Exploration

# 1) Replace non-numeric categorical values with numerics, ad defined below.
#       Zurich Class:   { C,...} = { 0, ...}
#       Spot Size:      { A,...} = { 0, ...}
#       Spot Distance:  { I,...} = { 0, ...}

# Insert logic to ....
#   1) search for a range of categorical values in each column,
#   2) create a sized numeric list based on the cardinality of the received
#   range,
#   3) For loop replace each categorical with a numeric equivalent
```

```
#TODO IF THERE IS TIME


###################################
# Organize data to be trained. Separate Features and Target variables.
#   'Edible' is the last column and the dataset's Target variable.
x = df.drop('C class', axis=1)   # Isolate the data features to train model.
y = df.get('C class')                    # Isolate the target variable

featureLabels = x.columns.values
targetLabel = 'C class'

###################################
# Setup and train the classifier trees based on different criterions.
#   Setting both too a maximum depth of four.

# 1) Entropy
entropy_Tree = tree.DecisionTreeClassifier(criterion='entropy',
                                           max_depth=4)
entropy_Tree = entropy_Tree.fit(x, y)   # Train the model based on features and
target

dot_entropy = tree.export_graphviz(entropy_Tree,
                                   out_file=None,
                                   feature_names=featureLabels,
                                   class_names=targetLabel,
                                   filled=True,
                                   rounded=True,
                                   special_characters=True)

# 2) Gini Index
gini_Tree = tree.DecisionTreeClassifier(criterion='gini',
                                        max_depth=4)

gini_Tree = gini_Tree.fit(x, y)   # Train the model based on features and
target

dot_gini = tree.export_graphviz(gini_Tree,
                                   out_file=None,
                                   feature_names=featureLabels,
                                   class_names=targetLabel,
                                   filled=True,
                                   rounded=True,
                                   special_characters=True)

# Generate Results
(graph_Entropy,) = pydot.graph_from_dot_data(dot_entropy)
graph_Entropy.write_png("FlareClassification-Entropy.png") #.png
```

```
(graph_Gini,) = pydot.graph_from_dot_data(dot_gini)
graph_Gini.write_png("FlareClassification-GiniIndex.png") #.png
```

Hand Work for section ‘Decision Tree 1: Hand Calculations’

Please see the external attached document.