9/28/2020 Untitled2 - Jupyter Notebook

In [1]:

**import** pandas **as** pd

df\_tennis **=** pd.read\_csv('PlayTennis.csv') print("\n Given Play Tennis Data Set:\n\n", df\_tennis)

Given Play Tennis Data Set:

PlayTennis Outlook Temperature Humidity Wind 0 No Sunny Hot High Weak 1 No Sunny Hot High Strong 2 Yes Overcast Hot High Weak 3 Yes Rainy Mild High Weak 4 Yes Rainy Cool Normal Weak 5 No Rainy Cool Normal Strong 6 Yes Overcast Cool Normal Strong 7 No Sunny Mild High Weak 8 Yes Sunny Cool Normal Weak 9 Yes Rainy Mild Normal Weak 10 Yes Sunny Mild Normal Strong 11 Yes Overcast Mild High Strong 12 Yes Overcast Hot Normal Weak 13 No Rainy Mild High Strong

In [2]:

df\_tennis.columns[0]

Out[2]:

'PlayTennis'

localhost:8888/notebooks/Untitled2.ipynb?kernel\_name=python3# 1/5

9/28/2020 Untitled2 - Jupyter Notebook

In [3]:

**def** entropy(probs):

**import** math

**return** sum( [**-**prob**\***math.log(prob, 2) **for** prob **in** probs] )

*#Function to calulate the entropy of the given Data Sets/List with respect to target attrib* **def** entropy\_of\_list(a\_list):

*#print("A-list",a\_list)*

**from** collections **import** Counter

cnt **=** Counter(x **for** x **in** a\_list) *# Counter calculates the propotion of class* print("\nClasses:",cnt)

*#print("No and Yes Classes:",a\_list.name,cnt)*

num\_instances **=** len(a\_list) *# = 14*

print("\n Number of Instances of the Current Sub Class is {0}:".format(num\_instances )) probs **=** [x **/** num\_instances **for** x **in** cnt.values()] *# x means no of YES/NO* print(probs)

print("\n Classes:",list(cnt.keys()))*#min(cnt),max(cnt))*

print(" \n Probabilities of Class {0} is {1}:".format(min(cnt),min(probs))) print(" \n Probabilities of Class {0} is {1}:".format(max(cnt),max(probs))) **return** entropy(probs) *# Call Entropy :*

*# The initial entropy of the YES/NO attribute for our dataset.*

print("\n INPUT DATA SET FOR ENTROPY CALCULATION:\n", df\_tennis['PlayTennis']) total\_entropy **=** entropy\_of\_list(df\_tennis['PlayTennis'])

print("\n Total Entropy of PlayTennis Data Set:",total\_entropy)

INPUT DATA SET FOR ENTROPY CALCULATION:

0 No

1 No

2 Yes

3 Yes

4 Yes

5 No

6 Yes

7 No

8 Yes

9 Yes

10 Yes

11 Yes

12 Yes

13 No

Name: PlayTennis, dtype: object

Classes: Counter({'Yes': 9, 'No': 5})

Number of Instances of the Current Sub Class is 14:

[0.35714285714285715, 0.6428571428571429]

Classes: ['No', 'Yes']

Probabilities of Class No is 0.35714285714285715:

Probabilities of Class Yes is 0.6428571428571429:

Total Entropy of PlayTennis Data Set: 0.9402859586706309

localhost:8888/notebooks/Untitled2.ipynb?kernel\_name=python3# 2/5

9/28/2020 Untitled2 - Jupyter Notebook

In [4]:

**def** information\_gain(df, split\_attribute\_name, target\_attribute\_name, trace**=**0): print("Information Gain Calculation of ",split\_attribute\_name)

'''

Takes a DataFrame of attributes, and quantifies the entropy of a target attribute after performing a split along the values of another attribute. '''

*# Split Data by Possible Vals of Attribute:*

df\_split **=** df.groupby(split\_attribute\_name)

print("split:",type(df\_split))

**for** name,group **in** df\_split:

print("Name:\n",name)

print("Group:\n",group)

*# Calculate Entropy for Target Attribute, as well as*

*# Proportion of Obs in Each Data-Split*

nobs **=** len(df.index)

print("NOBS",nobs)

*#define the aggregation based on target attribute name df\_agg\_ent=df\_ent\_prob* df\_ent\_prob **=** df\_split.agg({target\_attribute\_name : [entropy\_of\_list, **lambda** x: len(x)**/** *#print(target\_attribute\_name)*

*#print("df is",df\_agg\_ent)*

*#print(" Entropy List ",entropy\_of\_list)*

print(df\_ent\_prob.columns)

print("the entropy and the probability value for each attribute is",df\_ent\_prob)*#[targe* df\_ent\_prob.columns **=** ['Entropy', 'PropObservations']

*#if trace: # helps understand what fxn is doing:*

*# print(df\_agg\_ent)*

*# Calculate Information Gain:*

new\_entropy **=** sum( df\_ent\_prob['Entropy'] **\*** df\_ent\_prob['PropObservations'] ) overall\_entropy **=** entropy\_of\_list(df[target\_attribute\_name])

**return** overall\_entropy **-** new\_entropy

print('Info-gain for Outlook is :'**+**str( information\_gain(df\_tennis, 'Outlook', 'PlayTennis' print('\n Info-gain for Humidity is: ' **+** str( information\_gain(df\_tennis, 'Humidity', 'Play print('\n Info-gain for Wind is:' **+** str( information\_gain(df\_tennis, 'Wind', 'PlayTennis')) print('\n Info-gain for Temperature is:' **+** str( information\_gain(df\_tennis, 'Temperature','

y p y

3 Yes Rainy Mild High Weak

4 Yes Rainy Cool Normal Weak

5 No Rainy Cool Normal Strong

9 Yes Rainy Mild Normal Weak

13 No Rainy Mild High Strong

Name:

Sunny

Group:

PlayTennis Outlook Temperature Humidity Wind

0 No Sunny Hot High Weak

1 No Sunny Hot High Strong

7 No Sunny Mild High Weak

8 Yes Sunny Cool Normal Weak

10 Yes Sunny Mild Normal Strong

NOBS 14

Classes: Counter({'Yes': 4})

Number of Instances of the Current Sub Class is 4:

[1 0]

localhost:8888/notebooks/Untitled2.ipynb?kernel\_name=python3# 3/5

9/28/2020 Untitled2 - Jupyter Notebook

[1 0]

In [5]:

**def** id3(df, target\_attribute\_name, attribute\_names, default\_class**=None**):

*## Tally target attribute:*

**from** collections **import** Counter

cnt **=** Counter(x **for** x **in** df[target\_attribute\_name])*# class of YES /NO*

*## First check: Is this split of the dataset homogeneous?*

**if** len(cnt) **==** 1:

**return** next(iter(cnt)) *# next input data set, or raises StopIteration when EOF is*

*## Second check: Is this split of the dataset empty?*

*# if yes, return a default value*

**elif** df.empty **or** (**not** attribute\_names):

**return** default\_class *# Return None for Empty Data Set*

*## Otherwise: This dataset is ready to be devied up!*

**else**:

*# Get Default Value for next recursive call of this function:* default\_class **=** max(cnt.keys()) *#No of YES and NO Class*

*# Compute the Information Gain of the attributes:*

gainz **=** [information\_gain(df, attr, target\_attribute\_name) **for** attr **in** attribute\_na index\_of\_max **=** gainz.index(max(gainz)) *# Index of Best Attribute # Choose Best Attribute to split on:*

best\_attr **=** attribute\_names[index\_of\_max]

*# Create an empty tree, to be populated in a moment*

tree **=** {best\_attr:{}} *# Iniiate the tree with best attribute as a node* remaining\_attribute\_names **=** [i **for** i **in** attribute\_names **if** i **!=** best\_attr]

*# Split dataset*

*# On each split, recursively call this algorithm.*

*# populate the empty tree with subtrees, which*

*# are the result of the recursive call*

**for** attr\_val, data\_subset **in** df.groupby(best\_attr):

subtree **=** id3(data\_subset,

target\_attribute\_name,

remaining\_attribute\_names,

default\_class)

tree[best\_attr][attr\_val] **=** subtree

**return** tree

In [6]:

*# Get Predictor Names (all but 'class')*

attribute\_names **=** list(df\_tennis.columns)

print("List of Attributes:", attribute\_names)

attribute\_names.remove('PlayTennis') *#Remove the class attribute*

print("Predicting Attributes:", attribute\_names)

List of Attributes: ['PlayTennis', 'Outlook', 'Temperature', 'Humidity', 'Wi nd']

Predicting Attributes: ['Outlook', 'Temperature', 'Humidity', 'Wind']

localhost:8888/notebooks/Untitled2.ipynb?kernel\_name=python3# 4/5

9/28/2020 Untitled2 - Jupyter Notebook

In [7]:

**from** pprint **import** pprint

tree **=** id3(df\_tennis,'PlayTennis',attribute\_names) print("\n\nThe Resultant Decision Tree is :\n") *#print(tree)*

pprint(tree)

attribute **=** next(iter(tree))

print("Best Attribute :\n",attribute)

print("Tree Keys:\n",tree[attribute].keys())

Information Gain Calculation of Outlook

split: <class 'pandas.core.groupby.generic.DataFrameGroupBy'> Name:

Overcast

Group:

PlayTennis Outlook Temperature Humidity Wind 2 Yes Overcast Hot High Weak 6 Yes Overcast Cool Normal Strong 11 Yes Overcast Mild High Strong 12 Yes Overcast Hot Normal Weak Name:

Rainy

Group:

PlayTennis Outlook Temperature Humidity Wind 3 Yes Rainy Mild High Weak 4 Yes Rainy Cool Normal Weak 5 No Rainy Cool Normal Strong 9 Yes Rainy Mild Normal Weak 13 No Rainy Mild High Strong N

In [ ]:

localhost:8888/notebooks/Untitled2.ipynb?kernel\_name=python3# 5/5