
Exercise 9 – Decision Tree Construction: ID3 Algorithm

Date: 17/11/2022

Aim:

Deadline?	Is there a party?	Lazy?	Activity
Urgent	Yes	Yes	Party
Urgent	No	Yes	Study
Near	Yes	Yes	Party
None	Yes	No	Party
None	No	Yes	Pub
None	Yes	No	Party
Near	No	No	Study
Near	No	Yes	TV
Near	Yes	Yes	Party
Urgent	No	No	Study

To write functions for the following using Python:

1. Calculate Entropy and Information gain of the features (f) in F.
2. Read and store the dataset with appropriate data structure.
3. Construct the decision tree and derive the rules for every path from root to leaf node.

Code:

```
#Decision tree construction - ID3 algorithm

import pandas as pd
import numpy as np
import math
from collections import Counter

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

def entropy_of_list(a_list):
    cnt = Counter(x for x in a_list)   # Counter calculates the propotion of class
    num_instances = len(a_list)*1.0
    probs = [x / num_instances for x in cnt.values()]
    return entropy(probs)
```

```
def IG(df, split_attribute, target):
  tot_entropy = entropy_of_list(df[target])

  N = df.shape[0]
  df_split = df.groupby(split_attribute)
  entropy = []
  for name, group in df_split:
    temp = df.loc[df[split_attribute]== name]
    probability = temp.shape[0]/N
    entropy.append(probability*entropy_of_list(temp[target]))
  return tot_entropy - sum(entropy)

def id3(df, target, attribute_list, default_class = None):
  cnt = Counter(x for x in df[target])
  if len(cnt) == 1:
    print("\nFeature", next(iter(cnt)))
    print(df)
    return next(iter(cnt))
  if df.empty or len(attribute_list) == 0:
    return default_class
  else:
    default_class = max(cnt.keys())
    gainz = [IG(df, attr, target) for attr in attribute_list]
    max_IG = gainz.index(max(gainz))
    #print(gainz)
    best_attribute = attribute_list[max_IG]
    #print(best_attribute)
    tree = {best_attribute:{}}
    remaining_attributes = [i for i in attribute_list if i != best_attribute]
    for attr_val, df_subset in df.groupby(best_attribute):
      subtree = id3(df_subset, target, remaining_attributes, default_class)
      tree[best_attribute][attr_val] = subtree
    return tree

data = {
       'Deadline': ['Urgent', 'Urgent', 'Near', 'None', 'None', 'None', 'Near', 'Near', 'Near', 'Urgent'],
       'Party': ['Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'No'],
       'Lazy': ['Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No'],
       'Activity': ['Party', 'Study', 'Party', 'Party', 'Pub', 'Party', 'Study', 'TV', 'Party', 'Study']}

df = pd.DataFrame(data)
```

```
attribute_names = list(df.columns)
print("List of Attributes:", attribute_names)
attribute_names.remove('Activity') #Remove the class attribute
print("Predicting Attributes:", attribute_names)

from pprint import pprint
tree = id3(df,'Activity',attribute_names)
print("The Resultant Decision Tree is :\n")
pprint(tree)

attribute = next(iter(tree))
print("Best Attribute in the first level: ",attribute)
print("Keys on the first level: ",tree[attribute].keys())
```

Output:

```
List of Attributes: ['Deadline', 'Party', 'Lazy', 'Activity']
Predicting Attributes: ['Deadline', 'Party', 'Lazy']
```

```
Feature Study
  Deadline Party Lazy Activity
6     Near    No   No    Study

Feature TV
  Deadline Party Lazy Activity
7     Near    No  Yes       TV

Feature Pub
  Deadline Party Lazy Activity
4     None    No  Yes      Pub

Feature Study
  Deadline Party Lazy Activity
1   Urgent    No  Yes    Study
9   Urgent    No   No    Study

Feature Party
  Deadline Party Lazy Activity
0   Urgent   Yes  Yes    Party
2     Near   Yes  Yes    Party
3     None   Yes   No    Party
5     None   Yes   No    Party
8     Near   Yes  Yes    Party
```

```
The Resultant Decision Tree is :

{'Party': {'No': {'Deadline': {'Near': {'Lazy': {'No': 'Study', 'Yes': 'TV'}},
                               'None': 'Pub',
                               'Urgent': 'Study'}},
           'Yes': 'Party'}}
```

```
Best Attribute in the first level:  Party
Keys on the first level:  dict_keys(['No', 'Yes'])
```
