VIT-AP UNIVERSITY, ANDHRA PRADESH

Lab Sheet-10 : Decision Tree Classifier.

Academic year: 2022-2023 **Branch/ Class:** B.Tech
Semester: Fall **Date:**22/11/22
Faculty Name: Dr Aravapalli Rama Satish **School:** SCOPE

NAME: MAJJIGA JASWANTH REGNO:20BCD7171

****

1. a. Develop a python function to calculate Information Gain and Gain Ratio for a categorical attribute.
   Input: A data frame consists of Attribute and its Class Label
   Output: Splitting Criteria, Data Partitions after splitting, and corresponding calculated measures values.

   Code:

```
import pandas as pd
import numpy as np
```

```
[2] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df=pd.read_csv('tennis.csv')
df
```

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes
11	overcast	mild	high	True	yes
12	overcast	hot	normal	False	yes
13	rainy	mild	high	True	no

```
[3]  def compute_impurity(feature, impurity_criterion):
         probs = feature.value_counts(normalize=True)

         if impurity_criterion == 'entropy':
             impurity = -1 * np.sum(np.log2(probs) * probs)
         elif impurity_criterion == 'gini':
             impurity = 1 - np.sum(np.square(probs))
         else:
             raise ValueError('Unknown impurity criterion')

         return(round(impurity, 3))
```

```
[ ]  target_entropy = compute_impurity(df['play'], 'entropy')
     target_entropy

     0.94
```

b. Repeat (a) for every attribute of the following data set “tennis.csv”

```
def comp_feature_information_gain(df, target, descriptive_feature, split_criterion):

    print('target feature:', target)
    print('descriptive_feature:', descriptive_feature)
    print('split criterion:', split_criterion)

    target_entropy = compute_impurity(df[target], split_criterion)
    entropy_list = list()
    weight_list = list()
    for level in df[descriptive_feature].unique():
        df_feature_level = df[df[descriptive_feature] == level]
        entropy_level = compute_impurity(df_feature_level[target], split_criterion)
        entropy_list.append(round(entropy_level, 3))
        weight_level = len(df_feature_level) / len(df)
        weight_list.append(round(weight_level, 3))

    print('impurity of partitions:', entropy_list)
    print('weights of partitions:', weight_list)

    feature_remaining_impurity = np.sum(np.array(entropy_list) * np.array(weight_list))
    print('remaining impurity:', feature_remaining_impurity)

    information_gain = target_entropy - feature_remaining_impurity
    print('information gain:', information_gain)

    print('====================')

    return(information_gain)
```

```
for feature in df.drop(columns='play').columns:
    a= feature_info_gain = comp_feature_information_gain(df, 'play', feature, 'entropy')
```

```
target feature: play
descriptive_feature: outlook
split criterion: entropy
impurity of partitions: [0.971, -0.0, 0.971]
weights of partitions: [0.357, 0.286, 0.357]
remaining impurity: 0.693294
information gain: 0.24670599999999998
====================
target feature: play
descriptive_feature: temp
split criterion: entropy
impurity of partitions: [1.0, 0.918, 0.811]
weights of partitions: [0.286, 0.429, 0.286]
remaining impurity: 0.9117679999999999
information gain: 0.028232000000000035
====================
target feature: play
descriptive_feature: humidity
split criterion: entropy
impurity of partitions: [0.985, 0.592]
weights of partitions: [0.5, 0.5]
remaining impurity: 0.7885
information gain: 0.15149999999999997
====================
target feature: play
descriptive_feature: windy
split criterion: entropy
impurity of partitions: [0.811, 1.0]
weights of partitions: [0.571, 0.429]
remaining impurity: 0.8920809999999999
information gain: 0.047919000000000045
====================
```

c. Identify best attribute and draw the decision tree diagram up to 3 levels using anytree package and by applying iteratively step (b) on data subsets.

```
pip install anytree
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting anytree
  Downloading anytree-2.8.0-py2.py3-none-any.whl (41 kB)
     |████████████████████████████████| 41 kB 544 kB/s
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from anytree) (1.15.0)
Installing collected packages: anytree
Successfully installed anytree-2.8.0
```

```
from anytree import Node, RenderTree, find_by_attr
with open('input.txt', 'r') as f:
    lines = f.readlines()[1:]
    root = Node(lines[0].split(" ")[0])

    for line in lines:
        line = line.split(" ")
        Node("".join(line[1:]).strip(), parent=find_by_attr(root, line[0]))
    for pre, _, node in RenderTree(root):
        print("%s%s" % (pre, node.name))
```

```
outlook
├── sunny
│   └── humidity
│       ├── high
│       │   └── no
│       └── normal
│           └── yes
├── overcast
│   └── yes
└── rainy
    └── windy
        ├── true
        │   └── no
        └── false
            └── yes
```

d. Calculate accuracy of constructed Decision Tree.

```
[26] from sklearn.preprocessing import LabelEncoder
     A=LabelEncoder()
```

Loading...

```
Accuracy_Play=pd.read_csv('tennis.csv')
Accuracy_Play['outlook']=A.fit_transform(Accuracy_Play['outlook'])
Accuracy_Play['temp']=A.fit_transform(Accuracy_Play['temp'])
Accuracy_Play['humidity']=A.fit_transform(Accuracy_Play['humidity'])
Accuracy_Play['windy']=A.fit_transform(Accuracy_Play['windy'])
Accuracy_Play['play']=A.fit_transform(Accuracy_Play['play'])
```

```
[31] y=Accuracy_Play['play']
     x=Accuracy_Play.drop(['play'],axis=1)
```

```
[32] from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.30)
```

```
from sklearn.tree import DecisionTreeClassifier
classifier= DecisionTreeClassifier(criterion="entropy",random_state=100)
classifier.fit(x_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=100)
```

```
[35] y_pred=classifier.predict(x_test)
     import sklearn.metrics as metrics
     print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.8
```

```
import pandas as pd
import numpy as np
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df=pd.read_csv('tennis.csv')
df
```

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes
11	overcast	mild	high	True	yes
12	overcast	hot	normal	False	yes
13	rainy	mild	high	True	no

```
def compute_impurity(feature, impurity_criterion):
    probs = feature.value_counts(normalize=True)

    if impurity_criterion == 'entropy':
        impurity = -1 * np.sum(np.log2(probs) * probs)
    elif impurity_criterion == 'gini':
        impurity = 1 - np.sum(np.square(probs))
    else:
        raise ValueError('Unknown impurity criterion')

    return(round(impurity, 3))
```

```
target_entropy = compute_impurity(df['play'], 'entropy')
target_entropy
```

```
0.94
```

```
def comp_feature_information_gain(df, target, descriptive_feature, split_criterion):

    print('target feature:', target)
    print('descriptive_feature:', descriptive_feature)
    print('split criterion:', split_criterion)

    target_entropy = compute_impurity(df[target], split_criterion)
    entropy_list = list()
    weight_list = list()
    for level in df[descriptive_feature].unique():
        df_feature_level = df[df[descriptive_feature] == level]
        entropy_level = compute_impurity(df_feature_level[target], split_criterion)
        entropy_list.append(round(entropy_level, 3))
        weight_level = len(df_feature_level) / len(df)
        weight_list.append(round(weight_level, 3))

    print('impurity of partitions:', entropy_list)
    print('weights of partitions:', weight_list)

    feature_remaining_impurity = np.sum(np.array(entropy_list) * np.array(weight_list))
    print('remaining impurity:', feature_remaining_impurity)

    information_gain = target_entropy - feature_remaining_impurity
    print('information gain:', information_gain)

    print('====================')

    return(information_gain)


for feature in df.drop(columns='play').columns:
   a= feature_info_gain = comp_feature_information_gain(df, 'play', feature, 'entropy')
```

```
target feature: play
descriptive_feature: outlook
split criterion: entropy
impurity of partitions: [0.971, -0.0, 0.971]
weights of partitions: [0.357, 0.286, 0.357]
remaining impurity: 0.693294
information gain: 0.24670599999999998
====================
target feature: play
descriptive_feature: temp
split criterion: entropy
impurity of partitions: [1.0, 0.918, 0.811]
weights of partitions: [0.286, 0.429, 0.286]
remaining impurity: 0.9117679999999999
information gain: 0.028232000000000035
====================
target feature: play
descriptive_feature: humidity
split criterion: entropy
impurity of partitions: [0.985, 0.592]
```

```
weights of partitions: [0.5, 0.5]
remaining impurity: 0.7885
information gain: 0.15149999999999997
====================
target feature: play
descriptive_feature: windy
split criterion: entropy
impurity of partitions: [0.811, 1.0]
weights of partitions: [0.571, 0.429]
remaining impurity: 0.8920809999999999
information gain: 0.047919000000000045
====================
```

```
pip install anytree
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Collecting anytree
  Downloading anytree-2.8.0-py2.py3-none-any.whl (41 kB)
     |████████████████████████████████| 41 kB 544 kB/s
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (
Installing collected packages: anytree
Successfully installed anytree-2.8.0
```

```
from anytree import Node, RenderTree, find_by_attr
with open('input.txt', 'r') as f:
    lines = f.readlines()[1:]
    root = Node(lines[0].split(" ")[0])

    for line in lines:
        line = line.split(" ")
        Node("".join(line[1:]).strip(), parent=find_by_attr(root, line[0]))
    for pre, _, node in RenderTree(root):
        print("%s%s" % (pre, node.name))
```

```
outlook
├── sunny
│   └── humidity
│       ├── high
│       │   └── no
│       └── normal
│           └── yes
├── overcast
│   └── yes
└── rainy
    └── windy
        ├── true
        │   └── no
        └── false
            └── yes
```

```
from sklearn.preprocessing import LabelEncoder
A=LabelEncoder()
```

```
Accuracy_Play=pd.read_csv('tennis.csv')
Accuracy_Play['outlook']=A.fit_transform(Accuracy_Play['outlook'])
Accuracy_Play['temp']=A.fit_transform(Accuracy_Play['temp'])
Accuracy_Play['humidity']=A.fit_transform(Accuracy_Play['humidity'])
Accuracy_Play['windy']=A.fit_transform(Accuracy_Play['windy'])
Accuracy_Play['play']=A.fit_transform(Accuracy_Play['play'])


y=Accuracy_Play['play']
x=Accuracy_Play.drop(['play'],axis=1)


from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.30)


from sklearn.tree import DecisionTreeClassifier
classifier= DecisionTreeClassifier(criterion="entropy",random_state=100)
classifier.fit(x_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=100)
```

```
y_pred=classifier.predict(x_test)
import sklearn.metrics as metrics
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.8
```