Homework 3

In this homework assignment, you will implement a univariate feature selection method.

You will be given a toy dataset called 'Car Evaluation Data Set' (see: http://archive.ics.uci.edu/ml/datasets/Car+Evaluation (http://archive.ics.uci.edu/ml/datasets/Car+Evaluation) for details). You are not required to, but advised to test your code with the toy dataset, or any other dataset that contains categorical variables.

The given dataset contains six descriptive features and a target variable. Each of those are ordinal scale, categorical variables. The name of the target feature is 'evaluation'.

Note here that you are expected to write your own code, so DO NOT COPY AND PASTE CODE OR USE LIBRARY FUNCTIONS. The goal of the homework is not to see if you can call library functions but to have you practice with the impurity measures and feature selection techniques.

In [1]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import math
```

Briefing the Data Below -It's attributes and it's detail's

Read the dataset

In [2]:
```
edf = pd.read_csv('careval.csv')                          #Reading the data
print('\n')
print('Dataset Information')
print('\n')
display(edf.info())                                       #Information about the Data
print('\n')
print('Dataset Description ','\n')
print('\n')
display(edf.describe())                                   #Description(Count, unique, top and frequency of the attributes )
print('\n')
print('Top 5 observation of the Data ')
display(edf.head())                                       #Displaying top 5 observations
```

```
Dataset Information


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   buying      1728 non-null   object
 1   maint       1728 non-null   object
 2   doors       1728 non-null   object
 3   persons     1728 non-null   object
 4   lug_boot    1728 non-null   object
 5   safety      1728 non-null   object
 6   evaluation  1728 non-null   object
dtypes: object(7)
memory usage: 47.3+ KB

None


Dataset Description
```

	buying	maint	doors	persons	lug_boot	safety	evaluation
count	1728	1728	1728	1728	1728	1728	1728
unique	4	4	4	3	3	3	4
top	high	high	5more	more	big	high	unacc
freq	432	432	432	576	576	576	1210

```
Top 5 observation of the Data
```

	buying	maint	doors	persons	lug_boot	safety	evaluation
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

Calucating the Entropy values

You will create a method called IUFS (impurity-based univariate feature selection), which will select the most informative features with a univariate feature selection schema. This feature selection method will take the dataset, name of the target variable, number of features to be selected (k) and the measure of impurity as an input, and will output the names of k best features based on the information gain. You are expected to implement information gain, entropy and Gini index functions. Note here that this will be a univariate selection, which means that you need to test the features individually.

In [3]:
```python
# entropy (H)
def entropy(feature, dataset):
    values = dataset[feature].value_counts()
    Total=0
    for i in values:
        Total = Total + i;

    Total_Prob = 0
    for i in values:
        Indiviudal_Prob=(i/Total)*math.log(i/Total,2);             #Using the Entropy Formaule to calculate the Entropy
        Total_Prob = Total_Prob + Indiviudal_Prob
        Entropy = -round((Total_Prob),2)
    print('\n')
    return Entropy

for i in edf.columns:                                              #Printing the Entropy of each Attribute
    print('Entropy of ',i,' is ',entropy(i,edf))

#entropy('buying', edf)

#entropy('maint',edf)

# entropy('doors',edf)

# entropy('persons',edf)

# entropy('lug_boot',edf)

# entropy('safety',edf)

# entropy('evaluation',edf)
```

```
Entropy of  buying  is  2.0


Entropy of  maint  is  2.0


Entropy of  doors  is  2.0


Entropy of  persons  is  1.58


Entropy of  lug_boot  is  1.58


Entropy of  safety  is  1.58


Entropy of  evaluation  is  1.21
```

Calculating the gini values

In [4]:
```python
# gini index (Gini)


def gini(feature, dataset):
    values = dataset[feature].value_counts()
    Total=0
    for i in values:
        Total = Total + i;
    Total_Prob = 0
    Indiviudal_Prob=0
    for i in values:
        Indiviudal_Prob  = Indiviudal_Prob + (i/Total)*(i/Total);     #Formuale to Calculate Gini
    Total_Prob = round((1- Indiviudal_Prob),2)
#     print('Gini of ',feature,' is ',Total_Prob)
    print('\n')
    return Total_Prob

for i in edf.columns:                                                  #Displaying the Gini of the Attributes
    print('Gini of ',i,' is ',gini(i,edf))

gini('buying', edf)

# gini('maint',edf)

# gini('doors',edf)

# gini('persons',edf)

# gini('lug_boot',edf)

# gini('safety',edf)

# gini('evaluation',edf)
```

```
Gini of  buying  is  0.75


Gini of  maint  is  0.75


Gini of  doors  is  0.75


Gini of  persons  is  0.67


Gini of  lug_boot  is  0.67


Gini of  safety  is  0.67


Gini of  evaluation  is  0.46
```

Out[4]: 0.75

Calculating the Information Gain

In [10]:

```python
# information gain (IG)
import numpy as np
def InfoGain(feature, target, dataset, measure):

    if measure == "entropy":
        Target_Entropy = entropy(target,dataset)
        values,counts= np.unique(dataset[feature],return_counts=True)
        p_Entropy=0
        for i in range(len(values)):
                         # To find weighted gini we make usethe Gain_Dataset Gain_Dataset = dataset[dataset[feature] == values[i]]
            weighted_entropy = np.sum(counts[i]/np.sum(counts)*entropy(target,dataset[dataset[feature] == values[i]]))
            p_Entropy = p_Entropy + weighted_entropy
        print('\n')
        IG =  round(Target_Entropy - p_Entropy,6)
        print('IG from Entropy withrespect to attribute ',feature,' is ', IG)
        return IG

    else:

        Target_Gini = gini(target, dataset)
        values,counts= np.unique(dataset[feature],return_counts=True)
        probability_gini=0
        for i in range(len(values)):
           # To find weighted gini we make usethe Gain_Dataset Gain_Dataset = dataset[dataset[feature] == values[i]]
            weighted_Gini = (np.sum(counts[i]/np.sum(counts)*gini(target,dataset[dataset[feature] == values[i]])))
            probability_gini = probability_gini + weighted_Gini
        print('\n')
        IG =  round(Target_Gini - probability_gini,6)
        print('IG from gini withrespect to the attribute ',feature,' is ', IG)
        return IG



#Function calls to calculate InfoGain when measure of impurity is 'Entropy' and 'Gini' for buying attribute
InfoGain('buying','evaluation',edf,'entropy')
InfoGain('buying','evaluation',edf,'gini')

#Function calls to calculate InfoGain when measure of impurity is 'Entropy' and 'Gini' for safety attribute
InfoGain('safety','evaluation',edf,'entropy')
InfoGain('safety','evaluation',edf,'gini')
```

```
IG from Entropy withrespect to attribute  buying  is  0.1025
```

```
IG from gini withrespect to the attribute  buying  is  0.015
```

```
IG from Entropy withrespect to attribute  safety  is  0.263333
```

```
IG from gini withrespect to the attribute  safety  is  0.08
```

Out[10]:
```
0.08
```

Implementing IUFS function for finding k most informative features

In [6]:

```python
from pandas import DataFrame

def IUFS(target, dataset, k, measure='entropy'):
    if measure == 'entropy':
        column_names = ['Attribute', 'Info_Gain']
        Attribute = []
        Info_Gain = []
        Data = pd.DataFrame(columns = column_names)
        for i in edf.columns:
            Attribute.append(i)
            k1 = InfoGain(i,target,edf,'entropy')
            Info_Gain.append(k1)
        df = DataFrame(Attribute,columns = ['Attribute'])
        df.insert(1,'Info_Gain',Info_Gain,True)
        df=df[:-1]
        sorted_df = df.sort_values(by=['Info_Gain'],ascending=False)
        return sorted_df.head(k)
    else:
        column_names = ['Attribute', 'Info_Gain']
        Attribute = []
        Info_Gain = []
        Data = pd.DataFrame(columns = column_names)
        for i in edf.columns:
            Attribute.append(i)
            k1 = InfoGain(i,target,edf,'gini')
            Info_Gain.append(k1)
        df = DataFrame(Attribute,columns = ['Attribute'])
        df.insert(1,'Info_Gain',Info_Gain,True)
        df=df[:-1]
        sorted_df = df.sort_values(by=['Info_Gain'],ascending=False)
        return sorted_df.head(k)

Using_Entropy = IUFS('evaluation', edf, 2, measure='entropy')
display('Using Entropy as measure of impurity  k most informative features')
print('k most informative features are :',list(Using_Entropy.Attribute))
display(Using_Entropy)

Using_Gini = IUFS('evaluation', edf, 2, measure='gini')
display('Using gini as measure of impurity  k most informative features')
print('k most informative features are :',list(Using_Gini.Attribute))
display(Using_Gini)
```

IG from Entropy withrespect to attribute buying is 0.1025

IG from Entropy withrespect to attribute maint is 0.08

IG from Entropy withrespect to attribute doors is 0.01

IG from Entropy withrespect to attribute persons is 0.223333

IG from Entropy withrespect to attribute lug_boot is 0.033333

```
IG from Entropy withrespect to attribute  safety  is  0.263333
```

```
IG from Entropy withrespect to attribute  evaluation  is  1.21

'Using Entropy as measure of impurity  k most informative features'

k most informative features are : ['safety', 'persons']
```

	Attribute	Info_Gain
5	safety	0.263333
3	persons	0.223333

```
IG from gini withrespect to the attribute  buying  is  0.015
```

```
IG from gini withrespect to the attribute  maint  is  0.015
```

IG from gini withrespect to the attribute doors is 0.0075

IG from gini withrespect to the attribute persons is 0.073333

IG from gini withrespect to the attribute lug_boot is 0.006667

IG from gini withrespect to the attribute safety is 0.08

```
IG from gini withrespect to the attribute  evaluation  is  0.46

'Using gini as measure of impurity  k most informative features'

k most informative features are : ['safety', 'persons']
```

	Attribute	Info_Gain
5	safety	0.080000
3	persons	0.073333

Finding the Gain Ratio

Bonus

Improve the IUFS by including an option for gain ratio. Gain ratio is an alternative to information gain and can be used with either of the Gini index or entropy measures.

In [7]:
```
def GR(feature, target, dataset, measure):

    if measure == 'entropy':
        ratio = round(float(InfoGain(feature, target, dataset, measure))/float(entropy(feature,dataset)),5)
        #print('GR of attribute ',feature,'is', ratio)
        return ratio
    else:
        ratio = round(float(InfoGain(feature, target, dataset, measure))/float(gini(feature,dataset)),5)
        #print('GR of attribute ',feature,'is', ratio)
        return ratio

IG_of_buying_Entropy = GR('buying','evaluation', edf, 'entropy')
print('And its Gain Raito is',IG_of_buying_Entropy)
IG_of_buying = GR('buying','evaluation', edf, 'gini')
print('And its Gain Raito is',IG_of_buying)
```

```
IG from Entropy withrespect to attribute  buying  is  0.1025


And its Gain Raito is 0.05125
```

```
IG from gini withrespect to the attribute  buying  is  0.015


And its Gain Raito is 0.02
```

In [ ]:

Implemeting IUFS2

In [11]:

```python
from pandas import DataFrame

def IUFS2(target, dataset, k, measure,gain):
    if measure == 'entropy':
        if gain == 'IG':
            column_names = ['Attribute', 'Gain']
            Attribute = []
            Gain = []
            Data = pd.DataFrame(columns = column_names)
            for i in edf.columns:
                Attribute.append(i)
                k1 = InfoGain(i,target,edf,'entropy')
                Gain.append(k1)
            df = DataFrame(Attribute,columns = ['Attribute'])
            df.insert(1,'Gain',Gain,True)
            df=df[:-1]
            sorted_df = df.sort_values(by=['Gain'],ascending=False)
            return sorted_df.head(k)
        else:
            column_names = ['Attribute', 'Gain']
            Attribute = []
            Gain = []
            Data = pd.DataFrame(columns = column_names)
            for i in edf.columns:
                Attribute.append(i)
                k1 = GR(i,target,edf,'entropy')
                Gain.append(k1)
            df = DataFrame(Attribute,columns = ['Attribute'])
            df.insert(1,'Gain',Gain,True)
            df=df[:-1]
            sorted_df = df.sort_values(by=['Gain'],ascending=False)
            return sorted_df.head(k)

    else:
        if gain== 'IG':
            column_names = ['Attribute', 'Gain']
            Attribute = []
            Gain = []
            Data = pd.DataFrame(columns = column_names)
            for i in edf.columns:
                Attribute.append(i)
                k1 = InfoGain(i,target,edf,'gini')
                Gain.append(k1)
            df = DataFrame(Attribute,columns = ['Attribute'])
            df.insert(1,'Gain',Gain,True)
            df=df[:-1]
            sorted_df = df.sort_values(by=['Gain'],ascending=False)
            return sorted_df.head(k)
        else:
            column_names = ['Attribute', 'Gain']
            Attribute = []
            Gain = []
            Data = pd.DataFrame(columns = column_names)
            for i in edf.columns:
                Attribute.append(i)
                k1 = GR(i,target,edf,'gini')
                Gain.append(k1)
            df = DataFrame(Attribute,columns = ['Attribute'])
            df.insert(1,'Gain',Gain,True)
            df=df[:-1]
```

```
        sorted_df = df.sort_values(by=['Gain'],ascending=False)
        return sorted_df.head(k)


IUFS2_gini_GR = IUFS2('evaluation', edf, 3, measure='gini', gain='GR')
display(IUFS2_gini_GR)
display('K most informative features are : ',list(IUFS2_gini_GR.Attribute) )

IUFS2_entropy_IG = IUFS2('evaluation', edf, 2, measure='entropy', gain='IG')
display('K most informative features are : ',list(IUFS2_entropy_IG.Attribute) )
display(IUFS2_entropy_IG)
```

```
IG from gini withrespect to the attribute  buying  is  0.015
```

```
IG from gini withrespect to the attribute  maint  is  0.015
```

```
IG from gini withrespect to the attribute  doors  is  0.0075
```

```
IG from gini withrespect to the attribute  persons  is  0.073333
```

```
IG from gini withrespect to the attribute  lug_boot  is  0.006667
```

```
IG from gini withrespect to the attribute  safety  is  0.08
```

```
IG from gini withrespect to the attribute  evaluation  is  0.46
```

	Attribute	Gain
5	safety	0.11940
3	persons	0.10945
0	buying	0.02000

```
'K most informative features are : '

['safety', 'persons', 'buying']
```

IG from Entropy withrespect to attribute buying is 0.1025

IG from Entropy withrespect to attribute maint is 0.08

IG from Entropy withrespect to attribute doors is 0.01

IG from Entropy withrespect to attribute persons is 0.223333

IG from Entropy withrespect to attribute lug_boot is 0.033333

```
IG from Entropy withrespect to attribute  safety  is  0.263333
```

```
IG from Entropy withrespect to attribute  evaluation  is  1.21
'K most informative features are : '
['safety', 'persons']
```

	Attribute	Gain
5	safety	0.263333
3	persons	0.223333