Homework 2 - Classification

Name: < insert name here >

Remember that you are encouraged to discuss the problems with your instructors and classmates, but **you must write all code and solutions on your own**.

The rules to be followed for the assignment are:

- Do **NOT** load additional packages beyond what we've shared in the cells below.
- Some problems with code may be autograded. If we provide a function or class API **do not** change it.
- Do not change the location of the data or data directory. Use only relative paths to access the data.

In [1]:
```
import argparse
import pandas as pd
import numpy as np
import pickle
from pathlib import Path
from collections import defaultdict
```

[10 points] Problem 1 - Building a Decision Tree

A sample dataset has been provided to you in the './data/dataset.csv' path. Here are the attributes for the dataset. Use this dataset to test your functions.

- Age - ["<=30", "31-40", ">40"]
- Income - ["low", "medium", "high"]
- Student - ["no", "yes"]
- Credit Rating - ["fair", "excellent"]
- Loan - ["no", "yes"]

Note:

- A sample dataset to test your code has been provided in the location "data/dataset.csv". Please maintain this as it would be necessary while grading.
- Do not change the variable names of the returned values.
- After calculating each of those values, assign them to the corresponding value that is being returned.
- The "Loan" attribute should be used as the target variable while making calculations for your decision tree.

Loading [MathJax]/jax/output/HTML-CSS/jax.js

In [2]:
```
import math
import pandas as pd

def information_gain_target(dataset_file):

#        Input: dataset_file - A string variable which references the path
 to the dataset file.
#        Output: ig_loan - A floating point variable which holds the infor
mation gain associated with the target variable.
#
#        NOTE:
#        1. Return the information gain associated with the target variabl
e in the dataset.
#        2. The Loan attribute is the target variable
#        3. The pandas dataframe has the following attributes: Age, Income
, Student, Credit Rating, Loan
#        4. Perform your calculations for information gain and assign it t
o the variable ig_loan



    df = pd.read_csv(dataset_file)
    ig_loan = 0

    # your code here



    def entropy(class0, class1):
            if class0 == 0:
                return -(class1*math.log2(class1))
            elif class1 == 0:
                return -(class0*math.log2(class0))
            else:
                return -(class0 * math.log2(class0) + class1 * math.log2(c
lass1))
    class0 = df["Loan"].value_counts()[0] / len(df)
    class1 = df["Loan"].value_counts()[1] / len(df)
    s_entropy = entropy(class0, class1)
    ig_loan = s_entropy
    #print(ig_loan)
    return ig_loan







#ig_loan = information_gain_target('./data/dataset.csv')
#print(ig_loan)
#df = pd.read_csv('./data/dataset.csv')
#print(df)
#information_gain_target('./data/dataset.csv')
```

In [3]:
```
# This cell has hidden test cases that will run after you submit your assi
```

```
gnment.
```

In [4]:
```
attribute_values = {
    "Age": ["<=30", "31-40", ">40"],
    "Income": ["low", "medium", "high"],
    "Student": ["yes", "no"],
    "Credit Rating": ["fair", "excellent"]
}

attributes = ["Age", "Income", "Student", "Credit Rating"]
```

In [5]:
```
def information_gain(p_count_yes, p_count_no):

#   A helper function that returns the information gain when given counts
of number of yes and no values.
#   Please complete this function before you proceed to the information_ga
in_attributes function below.

    # your code here
    total_count = p_count_yes + p_count_no
    probs_yes = p_count_yes / total_count
    probs_no = p_count_no / total_count
    ig = -(probs_yes * math.log2(probs_yes) + probs_no * math.log2(probs_n
o))
    return ig
```

In [6]:
```
import operator

def information_gain_attributes(dataset_file, ig_loan, attributes, attribu
te_values):
    results = {
        "ig_attributes": {
            "Age": 0,
            "Income": 0,
            "Student": 0,
            "Credit Rating": 0
        },
        "best_attribute": ""
    }

    df = pd.read_csv(dataset_file)
    d_range = len(df)
    ig_loan = information_gain_target('./data/dataset.csv')


    for attribute in attributes: #["Age", "Income", "Student", "Credit Rat
ing"]
        ig_attribute = 0
        value_counts = dict()
        vcount = df[attribute].value_counts()
        weight_list = list()    # empty list every iteration when moving on
 to next attribute,
        entropy_list = list()  # save only the entropies of ith attribute
and then empty after done

        for att_value in attribute_values[attribute]:  #attribute_values =
```

```
 {"Age": ["<=30", "31-40", ">40"],

            # your code here
            df_feature_level = df[df[attribute] == att_value] # split into
 levels
            probs = df_feature_level["Loan"].value_counts()  / len(df_feat
ure_level["Loan"]) #attribute level based loan=yes/no proportion
            ent = -1 * np.sum(np.log2(probs)*probs)
            entropy_list.append(ent)
            weight = len(df_feature_level) / d_range
            weight_list.append(weight)


        ig_attribute = np.sum(np.array(entropy_list) * np.array(weight_lis
t))
        print("IG attribute:", ig_attribute)

        results["ig_attributes"][attribute] = ig_loan - ig_attribute
    results["best_attribute"] = max(results["ig_attributes"].items(), key=
operator.itemgetter(1))[0]
    return results



            #probsno = sum(df_feature_level[df_feature_level["Loan"] == "n
o"].value_counts()) / len(df_feature_level)
            #probsyes = sum(df_feature_level[df_feature_level["Loan"] == "
yes"].value_counts()) / len(df_feature_level)
            #ent = entropy(probsno, probsyes)
            #ig_attribute = ent
            #print(ig_attribute, attribute, att_value)

            #testing[testing["Loan"] == "yes"]
            #sum(df_feature_level[df_feature_level["Loan"] == "yes"].value
_counts())
        #print(weight_list)
        #results["ig_attributes"][attribute] = ig_loan - ig_attribute
        #Try referencing to information_gain for all proportion calculatio
ns

    results["best_attribute"] = max(results["ig_attributes"].items(), key=
operator.itemgetter(1))[0]
    return results
ig_loan = information_gain_target('./data/dataset.csv')
information_gain_attributes('./data/dataset.csv', ig_loan ,["Age", "Income
", "Student", "Credit Rating"], attribute_values = {
    "Age": ["<=30", "31-40", ">40"],
    "Income": ["low", "medium", "high"],
    "Student": ["yes", "no"],
    "Credit Rating": ["fair", "excellent"]
} )
```

```
IG attribute: 0.7378960810227786
IG attribute: 0.9674700395364009
IG attribute: 0.7841591278514218
IG attribute: 0.9080497460199801
```

Out[6]:

```
{'ig_attributes': {'Age': 0.2419726756283742,
  'Income': 0.012398717114751934,
  'Student': 0.19570962879973097,
  'Credit Rating': 0.07181901063117269},
 'best_attribute': 'Age'}
```

In [7]:
```
# This cell has hidden test cases that will run after you submit your assi
gnment.
```

[10 points] Problem 2 - Building a Naive Bayes Classifier

A sample dataset has been provided to you in the './data/dataset.csv' path. Here are the attributes for the dataset. Use this dataset to test your functions.

- Age - ["<=30", "31-40", ">40"]
- Income - ["low", "medium", "high"]
- Student - ["no", "yes"]
- Credit Rating - ["fair", "excellent"]
- Loan - ["no", "yes"]

Note:

- A sample dataset to test your code has been provided in the location "data/dataset.csv". Please maintain this as it would be necessary while grading.
- Do not change the variable names of the returned values.
- After calculating each of those values, assign them to the corresponding value that is being returned.
- The "Loan" attribute should be used as the target variable while making calculations for your naive bayes classifier.

In [11]:
```
from collections import defaultdict

def naive_bayes(dataset_file, attributes, attribute_values):

#   Input:
#       1. dataset_file - A string variable which references the path to t
he dataset file.
#       2. attributes - A python list which has all the attributes of the
dataset
#       3. attribute_values - A python dictionary representing the values
each attribute can hold.
#
#   Output: A proabbilities dictionary which contains the counts of when t
he Loan target variable is yes or no
#       depending on the input attribute.
#
#   Hint: Starter code has been provided to you to calculate the counts. Y
our code is very similar to the previous problem.

    probabilities = {
        "Age": { "<=30": {"yes": 0, "no": 0}, "31-40": {"yes": 0, "no": 0}
```

```
, ">40": {"yes": 0, "no": 0} },
        "Income": { "low": {"yes": 0, "no": 0}, "medium": {"yes": 0, "no":
 0}, "high": {"yes": 0, "no": 0}},
        "Student": { "yes": {"yes": 0, "no": 0}, "no": {"yes": 0, "no": 0}
 },
        "Credit Rating": { "fair": {"yes": 0, "no": 0}, "excellent": {"yes
": 0, "no": 0} },
        "Loan": {"yes": 0, "no": 0}
    }

    df = pd.read_csv(dataset_file)
    d_range = len(df)

    vcount = df["Loan"].value_counts() #total count of entries
    vcount_loan_yes = vcount["yes"]   #LOAN = yes and no in the entire df.
    vcount_loan_no = vcount["no"]

    probabilities["Loan"]["yes"] = vcount_loan_yes/d_range     # P(yes) in
 entire df
    probabilities["Loan"]["no"] = vcount_loan_no/d_range
    #print(probabilities["Age"]["<=30"]["yes"])
    for attribute in attributes:
        value_counts = dict()
        vcount = df[attribute].value_counts()
        for att_value in attribute_values[attribute]:

            # your code here

            #Formula is :::: P(att_value | yes) * P(yes) / P(att)


            df_feature_level = df[df[attribute] == att_value] #Split into
levels

            #P(attvalue|yes)
            att_yes = len(df_feature_level[df_feature_level["Loan"] == "ye
s"])   #all loan=yes for att_value
            att_no = len(df_feature_level[df_feature_level["Loan"] == "no"
])     #all loan=no for att_value
            p_att_yes = att_yes / len(df[df["Loan"] == "yes"])    #P(att_v
al | yes)
            p_att_no = att_no / len(df[df["Loan"] == "no"])     #P(att_val
 | no)
            print("all att_yes / all loan_yes:", p_att_yes)
            print("all att_no / all loan_no:", p_att_no)
            #P(att_value) = total number of attribute value / total number
 of data points


            p_att_value = len(df_feature_level) / d_range


            probs_yes = (p_att_yes * probabilities["Loan"]["yes"]) / p_att
_value
            probs_no = (p_att_no * probabilities["Loan"]["no"]) / p_att_va
lue
```

```
            #print(att_value, p_att_yes)
            #P(yes) is given above
            #P(att_yes) =
            #att_value_yes = total_yes / d_range
            #att_value_no = total_no / d_range

            probabilities[attribute][att_value]["yes"] = p_att_yes
            probabilities[attribute][att_value]["no"] = p_att_no

            #probs = df_feature_level[attribute[""]].value_counts()  / len
(df_feature_level["Loan"])
            #print(probs)
            #print(df_feature_level[df_feature_level["Loan"] == "no"])#.va
lue_counts()
            #probs = df_feature_level["Loan"].value_counts() / vcount
            #probabilities[attribute][att_value] =
            #print(probs["no"])
            #probabilities[attribute][att_value]["yes"] = probs / len(df_f
eature_level["Loan"])

    return probabilities
attribute_values = {
    "Age": ["<=30", "31-40", ">40"],
    "Income": ["low", "medium", "high"],
    "Student": ["yes", "no"],
    "Credit Rating": ["fair", "excellent"]
}

attributes = ["Age", "Income", "Student", "Credit Rating"]
naive_bayes('./data/dataset.csv',attributes, attribute_values)
```

```
all att_yes / all loan_yes: 0.2857142857142857
all att_no / all loan_no: 0.6
all att_yes / all loan_yes: 0.42857142857142855
all att_no / all loan_no: 0.0
all att_yes / all loan_yes: 0.2857142857142857
all att_no / all loan_no: 0.4
all att_yes / all loan_yes: 0.2857142857142857
all att_no / all loan_no: 0.2
all att_yes / all loan_yes: 0.42857142857142855
all att_no / all loan_no: 0.4
all att_yes / all loan_yes: 0.2857142857142857
all att_no / all loan_no: 0.4
all att_yes / all loan_yes: 0.7142857142857143
all att_no / all loan_no: 0.2
all att_yes / all loan_yes: 0.2857142857142857
all att_no / all loan_no: 0.8
all att_yes / all loan_yes: 0.7142857142857143
all att_no / all loan_no: 0.4
all att_yes / all loan_yes: 0.2857142857142857
all att_no / all loan_no: 0.6
```

Out[11]:
```
{'Age': {'<=30': {'yes': 0.2857142857142857, 'no': 0.6},
  '31-40': {'yes': 0.42857142857142855, 'no': 0.0},
  '>40': {'yes': 0.2857142857142857, 'no': 0.4}},
 'Income': {'low': {'yes': 0.2857142857142857, 'no': 0.2},
  'medium': {'yes': 0.42857142857142855, 'no': 0.4},
```

```
 'high': {'yes': 0.2857142857142857, 'no': 0.4}},
'Student': {'yes': {'yes': 0.7142857142857143, 'no': 0.2},
 'no': {'yes': 0.2857142857142857, 'no': 0.8}},
'Credit Rating': {'fair': {'yes': 0.7142857142857143, 'no': 0.4},
 'excellent': {'yes': 0.2857142857142857, 'no': 0.6}},
'Loan': {'yes': 0.5833333333333334, 'no': 0.4166666666666667}}
```

In [9]:
```
# This cell has hidden test cases that will run after you submit your assi
gnment.
```

In [ ]:
```

```