classification

December 10, 2021

0.1 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.

[45]:
```
import argparse
import pandas as pd
import numpy as np
import pickle
from pathlib import Path
from collections import defaultdict
```

[46]:
```
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␣
 ↪information gain associated with the target variable.
#
#       NOTE:
#       1. Return the information gain associated with the target variable 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 to the␣
 ↪variable ig_loan


    df = pd.read_csv(dataset_file)
    ig_loan = 0

    # your code here
    yes_perc = len(df[df['Loan'] == 'yes']) / len(df)
    no_perc = len(df[df['Loan'] == 'no']) / len(df)

    ig_loan = -(yes_perc) * np.log2(yes_perc) - (no_perc) * np.log2(no_perc)

    return ig_loan
```

[47]:
```
# This cell has hidden test cases that will run after you submit your␣
 ↪assignment.
```

[48]:
```
attribute_values = {
    "Age": ["<=30", "31-40", ">40"],
    "Income": ["low", "medium", "high"],
    "Student": ["yes", "no"],
    "Credit Rating": ["fair", "excellent"]
}

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

[49]:
```
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_gain_attributes function below.

    # your code here
    ig = 0
    total_count = p_count_yes + p_count_no
    p_count_yes_perc = p_count_yes / total_count
    p_count_no_perc = p_count_no / total_count

    if p_count_no_perc == 0:
        ig = -(p_count_yes_perc * np.log2(p_count_yes_perc))
    elif p_count_yes_perc == 0:
        ig = -(p_count_no_perc * np.log2(p_count_no_perc))
    else:
        ig = -(p_count_yes_perc * np.log2(p_count_yes_perc)) -␣
 ↪((p_count_no_perc) * np.log2(p_count_no_perc))
```

```
    return ig
```

0.1.1 [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.

[50]:
```
import operator

def information_gain_attributes(dataset_file, ig_loan, attributes,␣
 ↪attribute_values):

#        Input:
#            1. dataset_file - A string variable which references the path to␣
 ↪the dataset file.
#            2. ig_loan - A floating point variable representing the␣
 ↪information gain of the target variable "Loan".
#            3. attributes - A python list which has all the attributes of the␣
 ↪dataset
#            4. attribute_values - A python dictionary representing the values␣
 ↪each attribute can hold.
#
#        Output: results - A python dictionary representing the information␣
 ↪gain associated with each variable.
#            1. ig_attributes - A sub dictionary representing the information␣
 ↪gain for each attribute.
#            2. best_attribute - Returns the attribute which has the highest␣
 ↪information gain.
#
#        NOTE:
#        1. The Loan attribute is the target variable
```

```
#         2. The pandas dataframe has the following attributes: Age, Income,␣
↪Student, Credit Rating, Loan

    results = {
        "ig_attributes": {
            "Age": 0,
            "Income": 0,
            "Student": 0,
            "Credit Rating": 0
        },
        "best_attribute": ""
    }

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

    for attribute in attributes:
        ig_attribute = 0
        value_counts = dict()
        vcount = df[attribute].value_counts()
        for att_value in attribute_values[attribute]:

            # your code here
            value_counts[att_value] = vcount[att_value]
            yes_count = len(df[(df[attribute] == att_value) & (df['Loan'] ==␣
↪'yes')])
            no_count = len(df[(df[attribute] == att_value) & (df['Loan'] ==␣
↪'no')])
            ig_attribute += (value_counts[att_value] /d_range) *␣
↪information_gain(yes_count, no_count)

        results["ig_attributes"][attribute] = ig_loan - ig_attribute

    results["best_attribute"] = max(results["ig_attributes"].items(),␣
↪key=operator.itemgetter(1))[0]
    return results
```

[51]:
```
# This cell has hidden test cases that will run after you submit your␣
↪assignment.
```

0.1.2 [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.

[52]:
```
file_name = "data/dataset.csv"
attributes_ex = ['Age', 'Income', "Student", 'Credit Rating',]
attributes_val_ex = {
    'Age' : ["<=30", "31-40", ">40"],
    "Income" : ["low", "medium", "high"],
    "Student" : ["no", "yes"],
    "Credit Rating" : ["fair", "excellent"],
#     "Loan" : ["no", "yes"]
}
```

[53]:
```
from collections import defaultdict

def naive_bayes(dataset_file, attributes, attribute_values):

#   Input:
#       1. dataset_file - A string variable which references the path to the␣
 ↪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 the␣
 ↪Loan target variable is yes or no
#       depending on the input attribute.
#
#   Hint: Starter code has been provided to you to calculate the counts. Your␣
 ↪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()
    vcount_loan_yes = vcount["yes"]
    vcount_loan_no = vcount["no"]

    probabilities["Loan"]["yes"] = vcount_loan_yes/d_range
    probabilities["Loan"]["no"] = vcount_loan_no/d_range

    for attribute in attributes:
        value_counts = dict()
        vcount = df[attribute].value_counts()
        for att_value in attribute_values[attribute]:

            # your code here
            value_counts[att_value] = vcount[att_value]
            yes_count = len(df[(df[attribute] == att_value) & (df['Loan'] ==␣
↪'yes')])
            no_count = len(df[(df[attribute] == att_value) & (df['Loan'] ==␣
↪'no')])

            probabilities[attribute][att_value]["yes"] = yes_count /␣
↪vcount_loan_yes
            probabilities[attribute][att_value]["no"] = no_count /␣
↪vcount_loan_no


    return probabilities
```

[54]:
```
naive_bayes(file_name, attributes_ex, attributes_val_ex)
```

[54]:
```
{'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}}
```

[55]:
```
# This cell has hidden test cases that will run after you submit your␣
↪assignment.
```