CPS 844 Lab 6

03/23/2022
Section 1
Ryan Soliven
500840952

Part 1: Naïve Bayes Classifier

#1.

(0 points) Load the dataset weather.csv

Code

```
data = pd.read_csv("weather.csv")
```

#2

(15 points) As you may have noticed, the implementations of many classifiers do not support categorical data. Convert any categorical variable into dummy variables (a dummy variable takes only the value 0 or 1 to indicate the absence or presence of some attribute values). Hint: research about pandas' function get_dummies, and ensure the dummy variables are of type 'float'.

Code

```
data = pd.get_dummies(data).astype('float')
```

#3

(5 points) At this point, you should notice that the target attribute is now split into 2 target attributes. Drop the target attribute 'play_no'.

Code

```
data = data.drop(['play_no'],axis=1)
```

#4

(5 points) Continue with the preprocessing: separate the features attributes from the target attribute.

Code

```
classData = data['play_yes']
attributeData = data.drop(['play_yes'], axis=1)
```

#5

(5 points) Use the sklearn library to construct a Gaussian Naive Bayes classifier, then train this classifier.

Code

```
nb = GaussianNB()
nb = nb.fit(attributeData, classData)
proba = nb.predict_proba(attributeData)
```

#6

(15 points) Using the trained classifier, determine if the class label of ‘play_yes’ is more likely to be ‘0’ or ‘1’, for a “new day”, where the new day has the following attributes:

Outlook = sunny, Temperature = 66, Humidity = 90, Windy = true.

Code

```
data = pd.read_csv("weather.csv")
testData = [['sunny', 66, 90, True]]
testData = pd.DataFrame(testData, columns = ['outlook', 'temperature', 'humidity',
'windy'])
testData = pd.concat([data, testData], ignore_index=True)

testData = pd.get_dummies(testData).astype('float')
testData = testData.drop(['play_no'],axis=1)

testC = testData['play_yes']
testD = testData.drop(['play_yes'],axis=1)
proba = nb.predict_proba(testD)

print("Part 1 ----------------------\n")

if proba[-1][0] > proba[-1][1]:
    print("play_yes is more likely to be 0")
else:
    print("play_yes is more likely to be 1")
```

Results

play_yes is more likely to be 0

#7
(15 points) Given the weather of this “new day”, print out the likelihood of play = yes and the likelihood of play = no.

Code
```
print("Likelihood of play = yes: ", (proba[-1][1]))
print("Likelihood of play = no: ", (proba[-1][0]))
```

Results
Likelihood of play = yes: 8.492932894025e-05
Likelihood of play = no: 0.9999150706710601

Part 2: Association Analysis

#8
(0 points) Reload the dataset weather.csv if you have modified it.

Code
```
data = pd.read_csv("weather.csv")
```

#9
(15 points) Discretize the continuous data as follows: a. temperature: create 3 equal-width bins, where the original temperature values are replaced by the labels 'cool', 'mild' or, 'hot' b. humidity: create 2 equal-width bins, where the original humidity values are replaced by the labels 'normal' or 'high'

Code
```
data['temperature'] = pd.cut(x=data['temperature'], bins=3, labels=['cool','mild','hot'])
data['humidity'] = pd.cut(x=data['humidity'], bins=2, labels=['normal','high'])
```

#10
(10 points) Because of the implementation of the apyori library, you will also need to convert the boolean values from the attribute 'windy' to string. Hint: consider calling the method ‘map’ on the column ‘windy’ of your dataframe, in order to ‘map’ the boolean values to any string values.

Code
```
data['windy'] = data['windy'].map(str)
```

#11

(5 points) As seen during the lecture, the apyori library requires inputs in the form of a list of lists. Convert the whole dataset as a big list, where each ‘record’ is an inner list within the big list (you can re-use the code posted for the demo on D2L).

Code

```
weather = []
for i in range(data.shape[0]):
    weather.append(data.iloc[i].dropna().tolist())
```

#12

(5 points) You are ready to call the apriori function from the apyori module (you can reuse the code posted for the demo on D2L). You can start with a minimum support threshold of 0.28 and a minimum confidence threshold of 0.5.

Code

```
print("min_support = 0.28, min_confidence = 0.5")
for rule in rules:
    print(list(rule.ordered_statistics[0].items_base), '-->',
list(rule.ordered_statistics[0].items_add),
        'Support:',rule.support, 'Confidence:', rule.ordered_statistics[0].confidence )

rules = apriori(weather, min_support = 0.18, min_confidence = 0.3)
print("\nmin_support = 0.18, min_confidence = 0.3")
for rule in rules:
    print(list(rule.ordered_statistics[0].items_base), '-->',
list(rule.ordered_statistics[0].items_add),
        'Support:',rule.support, 'Confidence:', rule.ordered_statistics[0].confidence )

rules = apriori(weather, min_support = 0.4, min_confidence = 0.5)
print("\nmin_support = 0.4, min_confidence = 0.5")
for rule in rules:
    print(list(rule.ordered_statistics[0].items_base), '-->',
list(rule.ordered_statistics[0].items_add),
        'Support:',rule.support, 'Confidence:', rule.ordered_statistics[0].confidence )
```

Results

```
min_support = 0.28, min_confidence = 0.5
[] --> ['False'] Support: 0.5714285714285714 Confidence: 0.5714285714285714
[] --> ['high'] Support: 0.5 Confidence: 0.5
```

[] --> ['normal'] Support: 0.5 Confidence: 0.5
[] --> ['yes'] Support: 0.6428571428571429 Confidence: 0.6428571428571429
['False'] --> ['high'] Support: 0.2857142857142857 Confidence: 0.5
['False'] --> ['normal'] Support: 0.2857142857142857 Confidence: 0.5
['False'] --> ['yes'] Support: 0.42857142857142855 Confidence: 0.75
['cool'] --> ['normal'] Support: 0.2857142857142857 Confidence: 0.6666666666666666
['cool'] --> ['rainy'] Support: 0.2857142857142857 Confidence: 0.6666666666666666
['cool'] --> ['yes'] Support: 0.2857142857142857 Confidence: 0.6666666666666666
['high'] --> ['no'] Support: 0.2857142857142857 Confidence: 0.5714285714285714
['normal'] --> ['yes'] Support: 0.42857142857142855 Confidence: 0.8571428571428571
['overcast'] --> ['yes'] Support: 0.2857142857142857 Confidence: 1.0
['False'] --> ['normal', 'yes'] Support: 0.2857142857142857 Confidence: 0.5

min_support = 0.18, min_confidence = 0.3
[] --> ['False'] Support: 0.5714285714285714 Confidence: 0.5714285714285714
[] --> ['True'] Support: 0.42857142857142855 Confidence: 0.42857142857142855
[] --> ['cool'] Support: 0.42857142857142855 Confidence: 0.42857142857142855
[] --> ['high'] Support: 0.5 Confidence: 0.5
[] --> ['no'] Support: 0.35714285714285715 Confidence: 0.35714285714285715
[] --> ['normal'] Support: 0.5 Confidence: 0.5
[] --> ['rainy'] Support: 0.35714285714285715 Confidence: 0.35714285714285715
[] --> ['sunny'] Support: 0.35714285714285715 Confidence: 0.35714285714285715
[] --> ['yes'] Support: 0.6428571428571429 Confidence: 0.6428571428571429
['False'] --> ['cool'] Support: 0.21428571428571427 Confidence: 0.375
['False'] --> ['high'] Support: 0.2857142857142857 Confidence: 0.5
['False'] --> ['hot'] Support: 0.21428571428571427 Confidence: 0.375
['False'] --> ['normal'] Support: 0.2857142857142857 Confidence: 0.5
['False'] --> ['rainy'] Support: 0.21428571428571427 Confidence: 0.375
['False'] --> ['sunny'] Support: 0.21428571428571427 Confidence: 0.375
[] --> ['False', 'yes'] Support: 0.42857142857142855 Confidence:
0.42857142857142855
['True'] --> ['cool'] Support: 0.21428571428571427 Confidence: 0.5
['True'] --> ['high'] Support: 0.21428571428571427 Confidence: 0.5
['True'] --> ['no'] Support: 0.21428571428571427 Confidence: 0.5
['True'] --> ['normal'] Support: 0.21428571428571427 Confidence: 0.5
['True'] --> ['yes'] Support: 0.21428571428571427 Confidence: 0.5
['cool'] --> ['normal'] Support: 0.2857142857142857 Confidence: 0.6666666666666666
['cool'] --> ['rainy'] Support: 0.2857142857142857 Confidence: 0.6666666666666666
['cool'] --> ['yes'] Support: 0.2857142857142857 Confidence: 0.6666666666666666
['high'] --> ['hot'] Support: 0.21428571428571427 Confidence: 0.42857142857142855

['high'] --> ['no'] Support: 0.2857142857142857 Confidence: 0.5714285714285714
['high'] --> ['sunny'] Support: 0.21428571428571427 Confidence: 0.42857142857142855
['high'] --> ['yes'] Support: 0.21428571428571427 Confidence: 0.42857142857142855
['mild'] --> ['yes'] Support: 0.21428571428571427 Confidence: 0.75
['no'] --> ['sunny'] Support: 0.21428571428571427 Confidence: 0.6
['normal'] --> ['rainy'] Support: 0.21428571428571427 Confidence: 0.42857142857142855
[] --> ['normal', 'yes'] Support: 0.42857142857142855 Confidence: 0.42857142857142855
['overcast'] --> ['yes'] Support: 0.2857142857142857 Confidence: 1.0
['rainy'] --> ['yes'] Support: 0.21428571428571427 Confidence: 0.6
['False'] --> ['cool', 'yes'] Support: 0.21428571428571427 Confidence: 0.375
['False'] --> ['normal', 'yes'] Support: 0.2857142857142857 Confidence: 0.5
['False'] --> ['rainy', 'yes'] Support: 0.21428571428571427 Confidence: 0.375
['cool'] --> ['normal', 'yes'] Support: 0.21428571428571427 Confidence: 0.5
['high'] --> ['sunny', 'no'] Support: 0.21428571428571427 Confidence: 0.42857142857142855

min_support = 0.4, min_confidence = 0.5
[] --> ['False'] Support: 0.5714285714285714 Confidence: 0.5714285714285714
[] --> ['high'] Support: 0.5 Confidence: 0.5
[] --> ['normal'] Support: 0.5 Confidence: 0.5
[] --> ['yes'] Support: 0.6428571428571429 Confidence: 0.6428571428571429
['False'] --> ['yes'] Support: 0.42857142857142855 Confidence: 0.75
['normal'] --> ['yes'] Support: 0.42857142857142855 Confidence: 0.8571428571428571