Quiz3_NaiveBayesTest

November 8, 2018

In [1]:
```
import numpy as np
import pandas as pd
data = pd.read_csv("weatherX.csv")
data
```

Out[1]:

	Outlook	Temp	Humidity	Windy	Play
0	Sunny	hot	high	False	no
1	Sunny	hot	high	True	yes
2	Overcast	hot	high	False	no
3	Rainy	mild	high	False	no
4	Rainy	cool	normal	False	yes
5	Rainy	cool	normal	True	no
6	Overcast	cool	normal	True	no
7	Sunny	mild	high	False	no
8	Sunny	cool	normal	False	yes
9	Rainy	mild	normal	False	yes
10	Sunny	mild	normal	True	no
11	Overcast	mild	high	True	yes
12	Overcast	hot	normal	False	yes
13	Rainy	mild	high	True	yes

In [2]:
```
data.columns
```

Out[2]:
```
Index(['Outlook', 'Temp', 'Humidity', 'Windy', 'Play'], dtype='object')
```

In [3]:
```
data.index
```

Out[3]:
```
RangeIndex(start=0, stop=14, step=1)
```

In [4]:
```
data.iloc[:5][ ["Outlook", "Temp"] ]
```

Out[4]:

	Outlook	Temp
0	Sunny	hot
1	Sunny	hot
2	Overcast	hot
3	Rainy	mild
4	Rainy	cool

In [5]:
```
data.iloc[2:5][ ["Outlook", "Temp", "Play"] ]
```

```
Out[5]:     Outlook  Temp Play
        2  Overcast   hot   no
        3     Rainy  mild   no
        4     Rainy  cool  yes

In [6]: groupbyCount = data.groupby("Play").count()
        groupbyCount

Out[6]:       Outlook  Temp  Humidity  Windy
        Play
        no          7     7         7      7
        yes         7     7         7      7

In [7]: type( groupbyCount )

Out[7]: pandas.core.frame.DataFrame

In [8]: x=data.groupby("Play")["Play"].count()
        x

Out[8]: Play
        no     7
        yes    7
        Name: Play, dtype: int64

In [9]: type(x)

Out[9]: pandas.core.series.Series

In [10]: x/x.sum()

Out[10]: Play
         no     0.5
         yes    0.5
         Name: Play, dtype: float64

In [11]: print("Apriori Class Probabilies: Pr(Play=?)")
         x/x.sum()

Apriori Class Probabilies: Pr(Play=?)


Out[11]: Play
         no     0.5
         yes    0.5
         Name: Play, dtype: float64

In [12]: OutlookGivenPlay=data.groupby(['Play', 'Outlook'])['Play'].count()

         OutlookGivenPlay
```

```
Out[12]: Play  Outlook
         no    Overcast    2
               Rainy       2
               Sunny       3
         yes   Overcast    2
               Rainy       3
               Sunny       2
         Name: Play, dtype: int64

In [13]: OutlookGivenPlay.unstack()

Out[13]: Outlook  Overcast  Rainy  Sunny
         Play
         no              2      2      3
         yes             2      3      2

In [14]: type( OutlookGivenPlay.unstack() )

Out[14]: pandas.core.frame.DataFrame

In [15]: OutlookGivenPlay.unstack().sum(axis='columns')

Out[15]: Play
         no     7
         yes    7
         dtype: int64

In [16]: type( OutlookGivenPlay.unstack().sum(axis='columns') )

Out[16]: pandas.core.series.Series

In [17]: table= OutlookGivenPlay.unstack()
         subTotals = ( OutlookGivenPlay.unstack().sum(axis='columns') )
         subTotals

Out[17]: Play
         no     7
         yes    7
         dtype: int64

In [18]: table.divide(subTotals, axis='index')

Out[18]: Outlook  Overcast     Rainy     Sunny
         Play
         no       0.285714  0.285714  0.428571
         yes      0.285714  0.428571  0.285714

In [ ]: #In the following, let's use a loop to collect all the statistics for the Naive Bayes
```

In [19]:
```
#Problem #1

        print("Likelihood Prbabilities: Pr(Atribute=? | Play=?)")
        for column in data.columns:
            if (column != 'Play'):
                ###################################################
                # Write your own code to make the following happen
                ###################################################
```

```
Likelihood Prbabilities: Pr(Atribute=? | Play=?)
************************************************************
Attribute: Outlook
Pr( Outlook | Play)
Outlook  Overcast     Rainy     Sunny
Play
no       0.285714  0.285714  0.428571
yes      0.285714  0.428571  0.285714
************************************************************
Attribute: Temp
Pr( Temp | Play)
Temp      cool       hot      mild
Play
no    0.285714  0.285714  0.428571
yes   0.285714  0.285714  0.428571
************************************************************
Attribute: Humidity
Pr( Humidity | Play)
Humidity      high    normal
Play
no        0.571429  0.428571
yes       0.428571  0.571429
************************************************************
Attribute: Windy
Pr( Windy | Play)
Windy     False     True
Play
no     0.571429  0.428571
yes    0.571429  0.428571
```

In [20]:
```
#An equivalent but more elegant version
        print("Likelihood Prbabilities: Pr(Attribute=? | Play=?)")
        for column in data.columns[:-1]:
                ###################################################
                # reuse your own code in [19] above to make the following happen
                ###################################################
```

```
Likelihood Prbabilities: Pr(Attribute=? | Play=?)
************************************************************
Attribute: Outlook
Pr( Outlook | Play )
Outlook  Overcast     Rainy     Sunny
Play
no       0.285714  0.285714  0.428571
yes      0.285714  0.428571  0.285714
************************************************************
Attribute: Temp
Pr( Temp | Play )
Temp      cool       hot      mild
Play
no    0.285714  0.285714  0.428571
yes   0.285714  0.285714  0.428571
************************************************************
Attribute: Humidity
Pr( Humidity | Play )
Humidity      high    normal
Play
no        0.571429  0.428571
yes       0.428571  0.571429
************************************************************
Attribute: Windy
Pr( Windy | Play )
Windy     False     True
Play
no     0.571429  0.428571
yes    0.571429  0.428571
```

```
In [21]: #Problem #2

         #Generalize what you have in [11][19][20] to define a function for
         #   determining and printing the NaiveBayes model derived from data (a dataframe obje
         def printNaiveBayesModel(data):
                 ##################################################
                 # Write your own code to make the following in [22] happen
                 ##################################################
```

```
In [22]: printNaiveBayesModel(data)

Apriori Class Probabilies: Pr( Play =?)
==>
Play
no     0.5
yes    0.5
Name: Play, dtype: float64
```

```
Likelihood Prbabilities: Pr(Attribute=? | Play=?)
==>
************************************************************
Attribute: Outlook
Pr( Outlook | Play )
Outlook  Overcast     Rainy     Sunny
Play
no       0.285714  0.285714  0.428571
yes      0.285714  0.428571  0.285714
************************************************************
Attribute: Temp
Pr( Temp | Play )
Temp      cool       hot      mild
Play
no    0.285714  0.285714  0.428571
yes   0.285714  0.285714  0.428571
************************************************************
Attribute: Humidity
Pr( Humidity | Play )
Humidity      high    normal
Play
no        0.571429  0.428571
yes       0.428571  0.571429
************************************************************
Attribute: Windy
Pr( Windy | Play )
Windy     False     True
Play
no     0.571429  0.428571
yes    0.571429  0.428571
```

In [23]:
```
#Problem #3

        # Slighly modify what you have in [22] to define a function for
        #    determining and returning the NaiveBayes model derived from data (a dataframe ob
        # This function return should be a dictionary object where
        #    the keys are 'apriori' and the names of other attibutes,
        #    the corresponding value for 'apriori' is the series containing the apriori Class
        #    the corresponding value for an attribute name is the dataframe containing the li
        # in other words just like the prbability information shown in the output of [22]

        def getNaiveBayesModel(data):
            ################################################
            # Write your own code to make the following in [24]-[29] happen
            ################################################
```

```
In [24]: bayes = getNaiveBayesModel(data)
         bayes.keys()

Out[24]: dict_keys(['apriori', 'Outlook', 'Temp', 'Humidity', 'Windy'])

In [25]: type( bayes['Outlook'] )

Out[25]: pandas.core.frame.DataFrame

In [26]: bayes['Outlook']

Out[26]: Outlook  Overcast     Rainy     Sunny
         Play
         no       0.285714  0.285714  0.428571
         yes      0.285714  0.428571  0.285714

In [27]: bayes['Outlook'].loc['no']

Out[27]: Outlook
         Overcast    0.285714
         Rainy       0.285714
         Sunny       0.428571
         Name: no, dtype: float64

In [28]: bayes['Outlook']['Sunny'].loc['no']

Out[28]: 0.42857142857142855

In [29]: bayes['Windy'][True].loc['no']

Out[29]: 0.42857142857142855

In [30]: #Note that we can store a new case of observations as a dictionary
         observations = {'Outlook':'Sunny', 'Temp':'cool', 'Humidity':'high', 'Windy':True}

In [31]: observations.items()

Out[31]: dict_items([('Outlook', 'Sunny'), ('Temp', 'cool'), ('Humidity', 'high'), ('Windy', T

In [32]: for key in observations:
             print(key)

Outlook
Temp
Humidity
Windy


In [33]: for key,value in observations.items():
             print(key, value)
```

```
Outlook Sunny
Temp cool
Humidity high
Windy True
```

```
In [34]: bayes['Windy'][True].loc['no']
```

```
Out[34]: 0.42857142857142855
```

```
In [35]: for key,value in observations.items():
             print( bayes[key][value].loc['no'])
```

```
0.42857142857142855
0.2857142857142857
0.5714285714285714
0.42857142857142855
```

```
In [36]: #Problem #4
         #Write your code to demterine the likelihood of 'yes' versus 'no'
         for hypothesis in ['yes', 'no']:
           print('\nhypothesis=', hypothesis)
           ##################################################
           # Write your own code to make the following happen
           ##################################################
```

```
hypothesis= yes
0.2857142857142857
0.2857142857142857
0.42857142857142855
0.42857142857142855
posteriori= 0.007496876301541023

hypothesis= no
0.42857142857142855
0.2857142857142857
0.5714285714285714
0.42857142857142855
posteriori= 0.014993752603082045
```

```
In [37]: for hypothesis in bayes['apriori'].index:
             print( hypothesis )
```

```
no
yes
```

In [38]:
```
#Problem #5
#Generalize what you have in 4 into a function for priting out the likelihood of every
#  given a Bayes model and a case of observations

def printPrediction(bayes, observations):
  for hypothesis in bayes['apriori'].index:
    print('\nhypothesis=', hypothesis)
    ##################################################
    # Write your own code to make the output in [39] happen
    ##################################################
```

In [39]:
```
printPrediction(bayes, observations)
```

```
hypothesis= no
posteriori= 0.014993752603082045

hypothesis= yes
posteriori= 0.007496876301541023
```