Experiment-4

Manan Shah
2019130059
TE Comps
Batch C

Aim:
To train and test machine learning models using naive bayes algorithm.

Theory:
- The Bayes' Theorem is used to create a collection of classification algorithms known as Naive Bayes classifiers. It is a family of algorithms that share a similar idea, namely that each pair of features being classified is independent of the others.
- The Naive Bayes assumption is that each feature contributes equally and independently to the outcome.
- The Bayes' Theorem calculates the likelihood of an event occurring given the probability of a previous event.

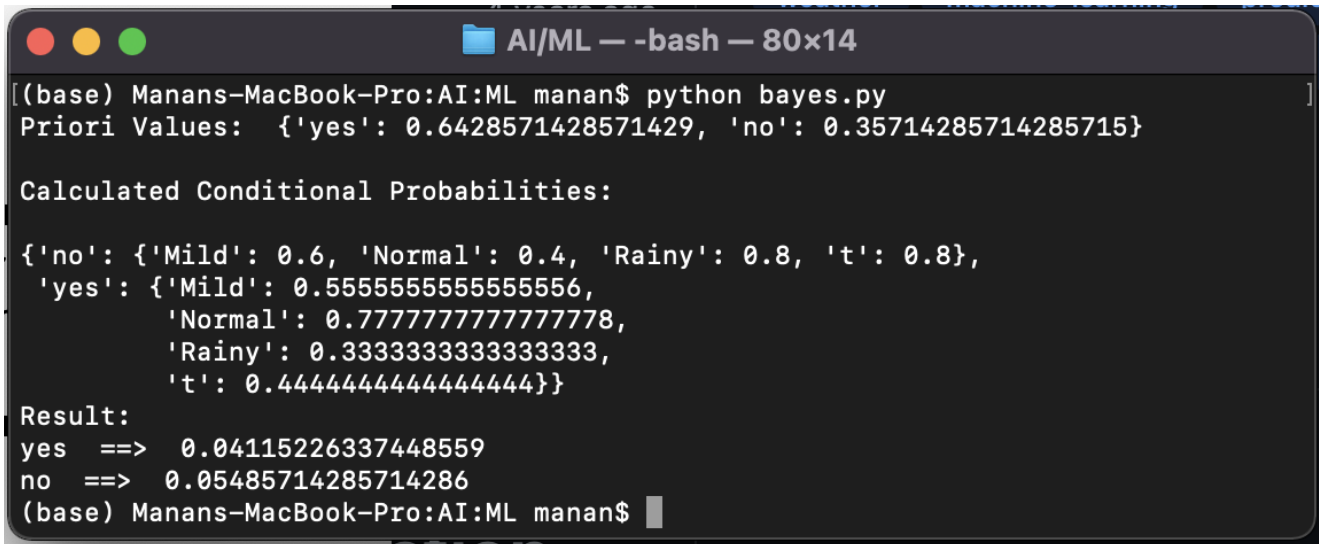
Code:

```
from functools import reduce

import pandas as pd
import pprint

class Classifier():
   data = None
   class_attr = None
   priori = {}
   cp = {}
   hypothesis = None


   def __init__(self,filename=None, class_attr=None ):
      self.data = pd.read_csv(filename, sep=',', header =(0))
      self.class_attr = class_attr

   '''
      probability(class) =    How many  times it appears in cloumn
                           __________________________________________
                             count of all class attribute
   '''
   def calculate_priori(self):
      class_values = list(set(self.data[self.class_attr]))
```

```
        class_data =  list(self.data[self.class_attr])
        for i in class_values:
            self.priori[i]  = class_data.count(i)/float(len(class_data))
        print ("Priori Values: ", self.priori)

    '''
        Here we calculate the individual probabilites
        P(outcome|evidence) =   P(Likelihood of Evidence) x Prior prob of outcome
                               ___________________________________________
                                                    P(Evidence)
    '''
    def get_cp(self, attr, attr_type, class_value):
        data_attr = list(self.data[attr])
        class_data = list(self.data[self.class_attr])
        total =1
        for i in range(0, len(data_attr)):
            if class_data[i] == class_value and data_attr[i] == attr_type:
                total+=1
        return total/float(class_data.count(class_value))

    '''
        Here we calculate Likelihood of Evidence and multiple all individual probabilities with priori
        (Outcome|Multiple Evidence) = P(Evidence1|Outcome) x P(Evidence2|outcome) x ... x P(EvidenceN|outcome) x
P(Outcome)
        scaled by P(Multiple Evidence)
    '''
    def calculate_conditional_probabilities(self, hypothesis):
        for i in self.priori:
            self.cp[i] = {}
            for j in hypothesis:
                self.cp[i].update({ hypothesis[j]: self.get_cp(j, hypothesis[j], i)})
        print ("\nCalculated Conditional Probabilities: \n")
        pprint.pprint(self.cp)

    def classify(self):
        print ("Result: ")
        for i in self.cp:
            print (i, " ==> ", reduce(lambda x, y: x*y, self.cp[i].values())*self.priori[i])

if __name__ == "__main__":
```

```
c = Classifier(filename="new_dataset.csv", class_attr="Play" )
c.calculate_priori()
c.hypothesis = {"Outlook":'Rainy', "Temp":"Mild", "Humidity":'Normal' , "Windy":'t'}

c.calculate_conditional_probabilities(c.hypothesis)
c.classify()
```

Output:



```
(base) Manans-MacBook-Pro:AI:ML manan$ python bayes.py
Priori Values:  {'yes': 0.6428571428571429, 'no': 0.35714285714285715}

Calculated Conditional Probabilities:

{'no': {'Mild': 0.6, 'Normal': 0.4, 'Rainy': 0.8, 't': 0.8},
 'yes': {'Mild': 0.5555555555555556,
         'Normal': 0.7777777777777778,
         'Rainy': 0.3333333333333333,
         't': 0.4444444444444444}}
Result:
yes  ==>  0.04115226337448559
no  ==>  0.05485714285714286
(base) Manans-MacBook-Pro:AI:ML manan$
```

Conclusion:

- The naive bayes experiment above taught me about the basic Bayes theorem. Bayes' theorem describes the probability of an event occurring in response to any condition. We calculate the probability of each output category in the naive bayes technique and choose the one with the highest probability.
- The naive bayes method is based on two assumptions: each data point in the dataset contributes to the dataset equally and independently.