Machine Learning and Pattern Recognition LAB
Chandra Prakash,
July-Dec 2018
LAB 2:

VIPASHA DHIMAN
04401032015
B.Tech(IT)

Bayesian Classifier :

Outlook	**Temprature**	**Humidity**	**Windy**	**Class**
sunny	hot	high	FALSE	n
sunny	hot	high	TRUE	n
overcast	hot	high	FALSE	p
rain	mild	high	FALSE	p
rain	cool	normal	FALSE	p
rain	cool	normal	TRUE	n
overcast	cool	normal	TRUE	p
sunny	mild	high	FALSE	n
sunny	cool	normal	FALSE	p
rain	mild	normal	FALSE	p
sunny	mild	normal	TRUE	p
overcast	mild	high	TRUE	p
overcast	hot	normal	FALSE	p
rain	mild	high	TRUE	n

Consider the following dataset with 4 attributes/features and two class(Play and not Play Tennis). Write a program to classify a new sample X when:

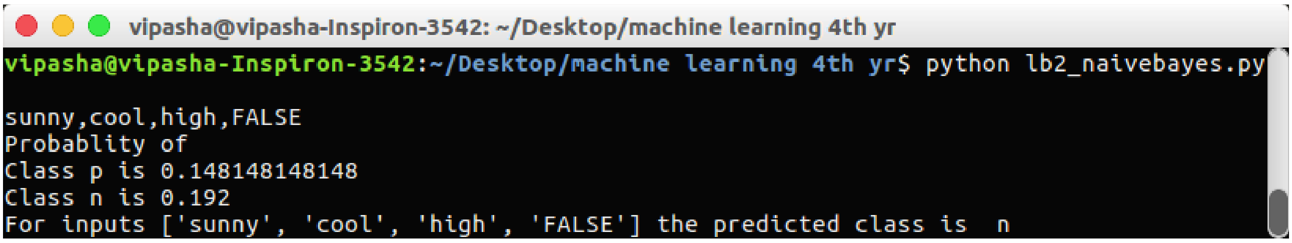
Solution:

```
import csv
import random

def loadCsv(filename):
        lines = csv.reader(open(filename, "rb"))
        dataset = list(lines)
        for i in range(len(dataset)):
                dataset[i] = [x for x in dataset[i]]
        return dataset
```

```
def splitDataset(dataset, splitRatio):
    trainSize = int(len(dataset) * splitRatio)
    trainSet = []
    copy = list(dataset)
    while len(trainSet) < trainSize:
        index = random.randrange(len(copy))
        trainSet.append(copy.pop(index))
    return [trainSet, copy]

def Diff(li1, li2):
   li_dif = [i for i in li1 + li2 if i not in li1 or i not in li2]
   return li_dif

def classesInDataset(dataset):
    classes = []
    for i in range(len(dataset)):
        vector=dataset[i]
        if vector[-1] not in classes:
            classes.append(vector[-1])
    return classes


def classesInDatasetWithFreq(dataset):
    classes = {}
    for i in range(len(dataset)):
        vector=dataset[i]
        if vector[-1] not in classes:
            classes[vector[-1]]=0
        classes[vector[-1]]+=1
    return classes


def separateByClassAndCalculateFrequency(dataset):
    separated = {}
    for i in range(len(dataset)):
        vector = dataset[i]
        if (vector[-1] not in separated):
            separated[vector[-1]] = {}
        for j in range(len(vector)-1):
            if vector[j] not in separated[vector[-1]]:
                separated[vector[-1]][vector[j]]=0
            separated[vector[-1]][vector[j]]+=1
    return separated


def calculateProbablity(dataset,classes,separated):
    visited={}
    cls=[]
```

```
	for i in range(len(dataset)):
		vector = dataset[i]
		if (vector[-1] not in visited):
			visited[vector[-1]] = []
			cls.append(vector[-1])
		for j in range(len(vector)-1):
			if vector[j] not in visited[vector[-1]]:
				separated[vector[-1]][vector[j]]=(separated[vector[-1]]
[vector[j]])/float(classes[vector[-1]])
				visited[vector[-1]].append(vector[j])
	leftFeatures=Diff(visited[cls[0]],visited[cls[1]])
	for i in range(len(leftFeatures)):
		for j in range(len(cls)):
			if leftFeatures[i] not in visited[cls[j]]:
				separated[cls[j]][leftFeatures[i]]=0.0
	return separated



def predict(model,check,classes,classesWithFrequency):
	max=0.0
	prediction='none'
	for i in range(len(classes)):
		probablity=1.0
		for j in range(len(check)):
			probablity*=model[classes[i]][check[j]]
		probablity*=classesWithFrequency[classes[i]]
		if probablity>max:
			max=probablity
			prediction=classes[i]
	return prediction


filename = 'np_dataset.csv'
dataset = loadCsv(filename)
trainingSet, testSet = splitDataset(dataset, 1.0)
classes=classesInDataset(trainingSet)
classesWithFrequency=classesInDatasetWithFreq(trainingSet)

frequency = separateByClassAndCalculateFrequency(trainingSet)

model=calculateProbablity(trainingSet,classesWithFrequency,frequency)


check=raw_input().split(",")

prediction=predict(model,check,classes,classesWithFrequency)
print 'For inputs', check ,'the predicted class is ',prediction
```

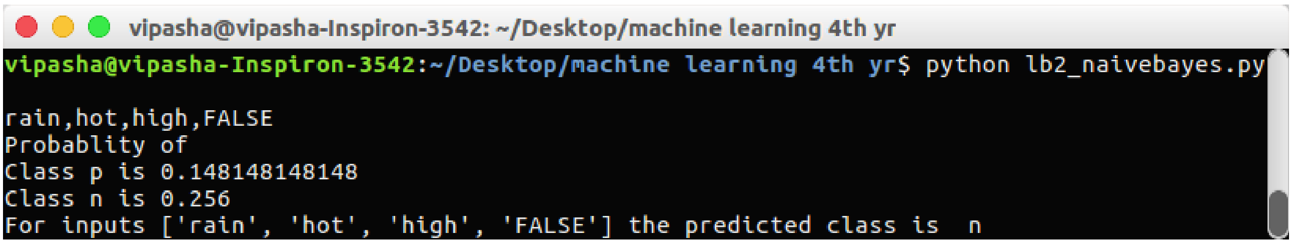
a)X=<sunny, cool, high, false>



vipasha@vipasha-Inspiron-3542: ~/Desktop/machine learning 4th yr

```
vipasha@vipasha-Inspiron-3542:~/Desktop/machine learning 4th yr$ python lb2_naivebayes.py

sunny,cool,high,FALSE
Probablity of
Class p is 0.148148148148
Class n is 0.192
For inputs ['sunny', 'cool', 'high', 'FALSE'] the predicted class is  n
```

b)X = <rain, hot, high, false>



vipasha@vipasha-Inspiron-3542: ~/Desktop/machine learning 4th yr

```
vipasha@vipasha-Inspiron-3542:~/Desktop/machine learning 4th yr$ python lb2_naivebayes.py

rain,hot,high,FALSE
Probablity of
Class p is 0.148148148148
Class n is 0.256
For inputs ['rain', 'hot', 'high', 'FALSE'] the predicted class is  n
```

OBSERVATIONS:

Naive Bayes classifiers, a family of classifiers that are based on the popular Bayes' probability theorem, are known for creating simple yet well performing models.

The formula I used in this is:

$$P(H/\text{Multiple Evidences}) = \frac{P(E1/H)* P(E2/H) \ldots *P(En/H) * P(H)}{P(\text{Multiple Evidences})}$$

Another important observation I noticed is that with the limited amount of dataset I was unable to train my classifier very well.

*We have two classes here i.e **n**: not play and **p**:play. We have 4 attributes which classify every set of input to a class.*
Attributes: Overlook, Temprature, Humidity and Windy