Assignment 1

**Ramya Tekumalla**

**PhD Student**

This is a submission of Assignment 1. Since I was unable to upload the code and csv file using icollege, I am uploading this word document which contains the code. For the code, I have used a csv file but I am uploading the excel file since that is the only accepted format.

I took the help of a tutorial to finish the code1 and the link to the tutorial is provided in the references. The dataset was prepared using the example given in the assignment2..

The dataset is split into two classes:

Class 0 – reviews with 1 star rating  
Class 1 – reviews with 5 star rating

The following 10 features were used in the dataset.

1. Battery Longevity
2. Product Quality
3. Sound Quality
4. Call Quality
5. Ease of Use
6. Value for Money
7. Comfort & Fit
8. Noise Cancellation
9. Sweat Proof
10. Connectivity

Comments are highlighted in red color and all the method names are self explanatory.

Code: nb\_classifier\_tekumalla.py

import csv

import random

import math

def loadCsv(filename): # loads the csv file to the program

lines = csv.reader(open(filename, "r", encoding='utf-8-sig'))

dataset = list(lines)

for i in range(len(dataset)):

dataset[i] = [float(x) for x in dataset[i]]

return dataset

def splitData(dataset, splitRatio): # randomly splits the dataset to train and test set

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 separateByClass(dataset):

separated = {}

for i in range(len(dataset)):

vector = dataset[i]

if (vector[-1] not in separated):

separated[vector[-1]] = []

separated[vector[-1]].append(vector)

return separated

def mean(numbers): # returns mean of the numbers

return sum(numbers)/float(len(numbers))

def stdev(numbers):

avg = mean(numbers)

variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers)-1)

return max(100,math.sqrt(variance))

def summarize(dataset):

summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(\*dataset)]

del summaries[-1]

return summaries

def summarizeByClass(dataset):

separated = separateByClass(dataset)

summaries = {}

for classValue, instances in separated.items():

summaries[classValue] = summarize(instances)

return summaries

def calculateProbability(x, mean, stdev): #calculateClassProbabilities() below, the probability of a given data instance is calculated by multiplying together the attribute probabilities for each class.

exponent = math.exp(-(math.pow(x-mean,2)/(2\*math.pow(stdev,2))))

return (1 / (math.sqrt(2\*math.pi) \* stdev)) \* exponent

def calculateClassProbabilities(summaries, inputVector):

probabilities = {}

for classValue, classSummaries in summaries.items():

probabilities[classValue] = 1

for i in range(len(classSummaries)):

mean, stdev = classSummaries[i]

x = inputVector[i]

probabilities[classValue] \*= calculateProbability(x, mean, stdev)

return probabilities

def predict(summaries, inputVector):

probabilities = calculateClassProbabilities(summaries, inputVector)

bestLabel, bestProb = None, -1

for classValue, probability in probabilities.items():

if bestLabel is None or probability > bestProb:

bestProb = probability

bestLabel = classValue

return bestLabel

def getPredictions(summaries, testSet):

predictions = []

for i in range(len(testSet)):

result = predict(summaries, testSet[i])

predictions.append(result)

return predictions

def getAccuracy(testSet, predictions):

correct = 0

for i in range(len(testSet)):

if testSet[i][-1] == predictions[i]:

correct += 1

return (correct/float(len(testSet))) \* 100.0

def main():

filename = 'dataset\_reviews\_features.csv'

splitRatio = 0.80

dataset = loadCsv(filename)

trainingSet, testSet = splitData(dataset, splitRatio)

print('Split {0} rows into train={1} and test={2} rows'.format(len(dataset), len(trainingSet), len(testSet)))

# prepare model

summaries = summarizeByClass(trainingSet)

# test model

predictions = getPredictions(summaries, testSet)

accuracy = getAccuracy(testSet, predictions)

print('Accuracy: {0}%'.format(accuracy))

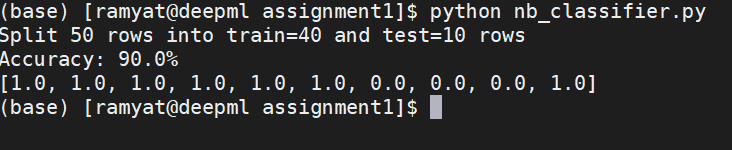
print(predictions)

main()

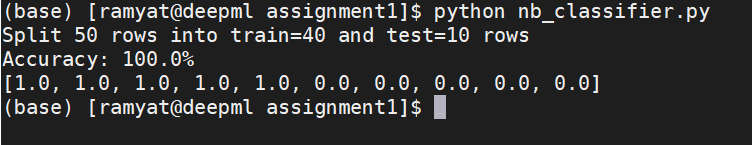
Result:

Since the training dataset is randomly created into train and test set, we will obtain different result each time we execute. I am attaching the screen shots of 3 results when executed 3 different times.

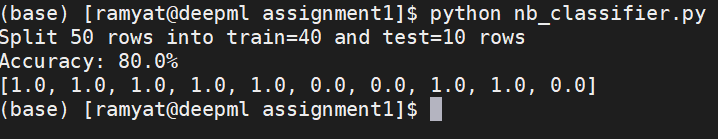
1. First try



1. Second try



1. Third try



References

1. <https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/>
2. <https://www.amazon.com/Bose-SoundSport-Wireless-Headphones-Aqua/dp/B01L7PWBRG/ref=sr_1_8?crid=3GW9D8LJITVRJ&keywords=bose+headphones&qid=1567607915&s=gateway&sprefix=bose+headphones%2Caps%2C174&sr=8-8>