Naive Bayes Classifier

Experiment Description

数据集

Most Popular Data Set中的wine数据集（对意大利同一地区声场的三种不同品种的酒做大量分析所得出的数据）

基本要求

a)采用分层采样的方式将数据集划分为训练集和测试集。

b)给定编写一个朴素贝叶斯分类器，对测试集进行预测，计算分类准确率。

中级要求

使用测试集评估模型，得到混淆矩阵，精度，召回率，F值。

高级要求

在中级要求的基础上画出三类数据的ROC曲线，并求出AUC值。

Experiment Process

1. Import libraries and dataset

```
import numpy as np
import math
import random
import csv
import operator

with open('wine.data') as csvfile:
     reader = csv.reader(csvfile)
     dataset = [row for row in reader]
```

2. Preprocess and spilt dataset as 7:3 into train dataset and test dataset. Data is stored in strings, so transform it into float. Use stratified sampling on train dataset. The numbers of three datasets are 42, 50, 33.

```
x_train = []
y_train = []
x_test = []
y_test = []
seed = []
train = random.sample(dataset[0: 59], 42)
train = train + random.sample(dataset[59: 59 + 71], 50)
train = train + random.sample(dataset[59 + 71: -1], 33)
test = [i for i in dataset if i not in train]

def to_float(dataset):
    y = []
    for i in dataset:
        for m in range(len(i)):
            i[m] = float(i[m])
        y.append(int(i[0]))
        i.pop(0)
    return dataset, y

x_train, y_train = to_float(train)
x_test, y_test = to_float(test)
```

3. Calculate probability density function.

   $P(c|x)$: posterior probability (given sample x, the probability of belonging to class c)

   $P(x|c)$: Assume the sample is in class c, the probability of observing sample x.

   $P(c)$: The prior probability of sample in class c. In reallife applications, the prior probability is usually unknown. It can only be asumed based on background knowledge, experiences, and training data. This is one of the difficulties of Bayes Classifier. Under this specific circumstance, $P(c)$ is assumed as normal distribution, with a probability density function of $\frac{1}{\sqrt{2\pi\sigma}}e^{-\frac{(x-u)^2}{2\sigma^2}}$.

```
def Bayes(data, p, avg, var):
    result = p
    for i in range(len(data)):
        result *=  1 / (math.sqrt(2 * math.pi * var[i])) * math.exp(-((data[i] - 
avg[i])**2) / (2 * var[i]))
    return result
```

4. Classification

   By calculating posterior probability, classify test dataset, and calculate the accuracy.

```
def classifier(x_train, x_test):
    result = []
    x_train = np.array(x_train)
    avg1 = x_train[:42].mean(axis = 0)
    var1 = x_train[:42].var(axis = 0)
    avg2 = x_train[42 : 42 + 50].mean(axis = 0)
    var2 = x_train[42 : 42 + 50].var(axis = 0)
    avg3 = x_train[42 + 50 : ].mean(axis = 0)
    var3 = x_train[42 + 50 : ].var(axis = 0)
    for i in range(len(x_test)):
        temp = 1
        max = Bayes(x_test[i], 59 / (59 + 71 + 48), avg1, var1)
        if Bayes(x_test[i], 71 / (59 + 71 + 48), avg2, var2) > max:
            temp = 2
            max = Bayes(x_test[i], 71 / (59 + 71 + 48), avg2, var2)
        if Bayes(x_test[i], 48 / (59 + 71 + 48), avg3, var3) > max:
            temp = 3
        result.append(temp)
    return result
```

```
def simrate(ls1, ls2):
    num = 0
    l = len(ls1)
    for i in range(l):
        if ls1[i] == ls2[i]:
            num += 1
    return format(num / l, '.2%')

predict = classifier(x_train, x_test)

print("分类的准确率是", simrate(predict, y_test))
```

5. Generate confusion matrix

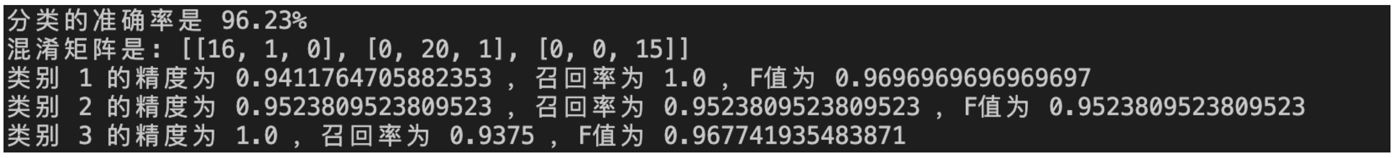
```
   def confuse_maxtria(predict, fact):
       ls = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
       for i in range(len(predict)):
           ls[fact[i] - 1][predict[i] - 1] += 1
       return ls

   print("混淆矩阵是:", confuse_maxtria(predict, y_test))
```

6. Calculate precision, recall and F-measure
    - True Positive: Predict positive values as positive class
    - True Negative: Predict negative values as negative class
    - False Positive: Predict negative values as positive class (Type I error)
    - False Negative: Predict positive values as negative class (Type II error)
    - Precision: $p = \frac{TP}{TP+FP}$
    - Recall: $p = \frac{TP}{TP+FN}$
    - F-measure: $F - Measure = \frac{(a^2+1)P*R}{P+R}$，本次实验中$a$值取为1，a>1 recall weight more; a<1 precision weight more

```
   def get_feature(confuse_maxtria):
       for index in range(len(confuse_maxtria)):
           truth = confuse_maxtria[index][index]
           total = 0
           total2 = 0
           for i in range(len(confuse_maxtria)):
               total += confuse_maxtria[index][i]
           for i in range(len(confuse_maxtria)):
               total2 += confuse_maxtria[i][index]
           precision = truth / total
           recall = truth / total2
           f_rate = 2 * precision * recall / (precision + recall)
           print("类别", index + 1, "的精度为", precision, ", 召回率为", recall, ", F值
为", f_rate)
```

```
    get_feature(confuse_maxtria(predict, y_test))
```

Experiment Result



```
分类的准确率是 96.23%
混淆矩阵是: [[16, 1, 0], [0, 20, 1], [0, 0, 15]]
类别 1 的精度为 0.9411764705882353 , 召回率为 1.0 , F值为 0.9696969696969697
类别 2 的精度为 0.9523809523809523 , 召回率为 0.9523809523809523 , F值为 0.9523809523809523
类别 3 的精度为 1.0 , 召回率为 0.9375 , F值为 0.967741935483871
```

It can be seen that this experiment have a high accuracy, and the evaluation indexes(precision, recall and F-measure) are good. The classification outcome is ideal.