## **广州大学学生实验报告**

**开课学院及实验室：**计算机科学与工程实验室 **2021年11月15日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | **计算机科学与网络工程学院** | **年级/专业/班** | **网络193** | **姓名** | 吴伟俊 | **学号** | 1906200107 |
| **实验课程名称** | **人工智能导论实验** | | | | | **成绩** |  |
| **实验项目名称** | **分类算法** | | | | | **指导老师** | 张少宏 |

(\*\*\*报告只能为文字和图片,老师评语将添加到此处,学生请勿作答\*\*\*)

**一、实验内容**

**问题描述**：分类算法是解决分类问题的方法，是数据挖掘、机器学习和模式识别中一个重要的研究领域。分类算法通过对已知类别训练集的分析，从中发现分类规则，以此预测新数据的类别。分类算法的应用非常广泛，银行中风险评估、客户类别分类、文本检索和搜索引擎分类、安全领域中的入侵检测以及软件项目中的应用等等。

**内容提要**：针对教师指定的两类公用数据集（纯数值型例如UCIIris, 混杂型数据例如UCI Bank Marketing），学生至少实现两种分类算法，并比较分析结果原因。本次实验主要内容包括数据处理、算法实现和评价方法。鼓励与其他方法尤其是业界领先算法进行比较分析，鼓励创新设计求解方法。

**二、实验设备**

1. 实验设备：计算机；

2. 平台：Windows操作系统，vscode

**三、实验步骤**

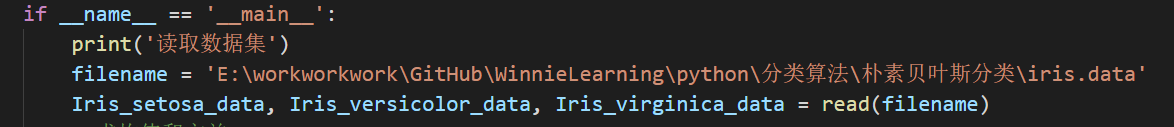
* 读取数据，并做预处理。
* 至少实现两种分类算法，选择评价方法比较结果并分析原因
* 选择适当可视化方法显示结果。
* 扩展选做题：分析考虑数据的特性并和具体分类方法的匹配。

**四、分析说明**

* 朴素贝叶斯分类算法

使用概率统计的知识对样本数据集进行分类。给定一个属性值，其属于某个类的概率叫做条件概率。对于一个给定的类值，将每个属性的条件概率相乘，便得到一个数据样本属于某个类的概率。

因此，使用改算法对iris.data数据集进行分类。据作为测试数据集。



处理数据，求取均值和方差。

文本

描述已自动生成

计算先验概率

文本

描述已自动生成

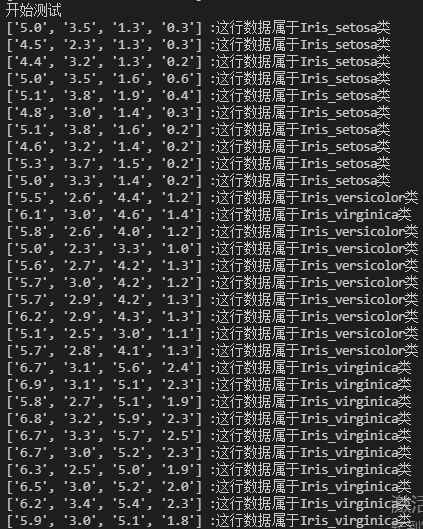
最后对结果进行判断，并进行可视化：

文本

描述已自动生成

结果：

背景图案

描述已自动生成 

发现准确率100%

* Knn算法

Knn算法的主要思路是：如果一个样本在特征空间中的K个最相似的样本中的大多数属于某一个类别，则该样本也属于这个类别。这是一个将数据集合中每一个记录进行分类的方法。所以可对记录特征多的数据集进行分类，因此这里对bank-additional数据集进行分类。

数据预处理和对数据进行划分，分出测试数据集：

文本

描述已自动生成

计算处欧氏距离，找出距离最近的k个实例，获取距离最近的k个实例中占比并进行分类：

文本

描述已自动生成

可视化：

文本

描述已自动生成

结果：

文本

中度可信度描述已自动生成

图表, 散点图

描述已自动生成 图形用户界面, 文本

描述已自动生成

图形用户界面, 文本

描述已自动生成

**实验总结：**

本次实验我使用两种不同的分类算法：朴素贝叶斯和knn来对数据集进行分类，了解了不同分类算法的原理和优缺点. 朴素贝叶斯准确率高，但是适用性低，只适合少特征的数据分类。Knn准确率相对较低，但是实用性广，能适应现实中多特征的数据分类。

**函数源代码如下：**

**bayes\_iris.py:**

import math

Iris\_setosa\_data = []

Iris\_versicolor\_data = []

Iris\_virginica\_data = []

test\_data = []

def split(data, l, r):

    s = []

    for line in data[l : r - 10]:

        line = line.strip().split(',')

        s.append(line[0 : 4])

    for line in data[r - 10 : r]:

        line = line.strip().split(',')

        test\_data.append(line[0 : 4])

    return s

def read(filename):

    # 读取训练数据集前40条数据，剩下的10条数据作为测试数据集

    f = open(filename, 'r')

    all\_lines = f.readlines()

    Iris\_setosa\_data = split(all\_lines, 0, 50)

    Iris\_versicolor\_data = split(all\_lines, 50, 100)

    Iris\_virginica\_data = split(all\_lines, 100, 150)

    return Iris\_setosa\_data, Iris\_versicolor\_data, Iris\_virginica\_data

def calc(train\_data):

    # 计算均值和方差

    sum\_x = [0.0, 0.0, 0.0, 0.0]

    for x in train\_data:

        for id in range(0, 4):

            sum\_x[id] += float(x[id])

    ux = []

    for id in range(0, 4):

        ux.append(sum\_x[id] / 45)

    k = [0.0, 0.0, 0.0, 0.0]

    for x in train\_data:

        for id in range(0, 4):

            k[id] += (float(x[id]) - ux[id]) \*\* 2

    vx = []

    for id in range(0, 4):

        vx.append(k[id] / 45)

    return ux, vx

def calc\_P(ux, vx, data):

    # 计算每个属性估计条件概率

    px = []

    C = 1 / math.sqrt(2 \* math.pi)

    for id in range(0, 4):

        px.append(C \* math.exp(-(float(data[id]) - ux[id]) \*\* 2 / (2 \* vx[id])))

    return px

if \_\_name\_\_ == '\_\_main\_\_':

    print('读取数据集')

    filename = 'E:\workworkwork\GitHub\WinnieLearning\python\分类算法\朴素贝叶斯分类\iris.data'

    Iris\_setosa\_data, Iris\_versicolor\_data, Iris\_virginica\_data = read(filename)

    # 求均值和方差

    Iris\_setosa\_ux, Iris\_setosa\_vx = calc(Iris\_setosa\_data)

    Iris\_versicolor\_ux, Iris\_versicolor\_vx = calc(Iris\_versicolor\_data)

    Iris\_virginica\_ux, Iris\_virginica\_vx = calc(Iris\_virginica\_data)

    print ('测试数据')

    for x in test\_data:

        print(x)

    print('开始测试')

    # 估计类先验概率

    sum\_data = len(Iris\_versicolor\_data) + len(Iris\_virginica\_data) + len(Iris\_setosa\_data)

    p = []

    for data in [len(Iris\_setosa\_data), len(Iris\_versicolor\_data), len(Iris\_virginica\_data)]:

        p.append(data / sum\_data)

    for data in test\_data:

        Iris\_setosa\_Px = calc\_P(Iris\_setosa\_ux, Iris\_setosa\_vx, data)

        Iris\_versicolor\_Px = calc\_P(Iris\_versicolor\_ux, Iris\_versicolor\_vx, data)

        Iris\_virginica\_Px = calc\_P(Iris\_virginica\_ux, Iris\_virginica\_vx, data)

        P = [p[0], p[1], p[2]] # Iris\_setosa\_P, Iris\_versicolor\_P, Iris\_virginica\_P

        for id in range(0, 4):

            P[0], P[1], P[2] = P[0] \* Iris\_setosa\_Px[id], P[1] \* Iris\_versicolor\_Px[id], P[2] \* Iris\_virginica\_Px[id]

        if P[0] == max(P):

            print(data, ":这行数据属于Iris\_setosa类")

        elif P[1] == max(P):

            print(data, ":这行数据属于Iris\_versicolor类")

        else:

            print(data, ":这行数据属于Iris\_virginica类")

**knn.py:**

import pandas as pd

import random

import math

import operator

import matplotlib.pyplot as plt

def change\_job(state):

    # 转换职业状态

    job = [ "admin.","blue-collar","entrepreneur","housemaid","management","retired",

            "self-employed","services","student","technician","unemployed","unknown"]

    for i in range(len(job)):

        if state == job[i]:

            return i

def change\_marital(state):

    # 转换婚姻状态:未知：0,单身：1,结婚：2,离婚：3

    marital = ["divorced","married","single","unknown"]

    for i in range(len(marital)):

        if state == marital[i]:

            return i

def change\_education(state):

    # 转换教育水平

    education = ["basic.4y","basic.6y","basic.9y","high.school","illiterate","professional.course","university.degree","unknown"]

    for i in range(len(education)):

        if state == education[i]:

            return i

def change\_binary(state):

    # 二分类转换

    if state == 'yes':

        state = 1

    elif state == 'no':

        state = 0

    return state

def change\_contact(state):

    # 转换接触方式: 未知(unknown):0、网络(cellular)：1、电话（telephone）：2

    contact = ['unknown', 'cellular', 'telephone']

    for i in range(len(contact)):

        if state == contact[i]:

            return i

def change\_poutcome(state):

    # 转换poutcome（上次成功与否）

    poutcome = ["failure","nonexistent","success"]

    for i in range(len(poutcome)):

        if state == poutcome[i]:

            return i

def change\_month(state):

    # 转换month

    month = ['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec']

    for i in range(len(month)):

        if state == month[i]:

            return i

def change\_week(state):

    # 转换week

    week = ["mon","tue","wed","thu","fri"]

    for i in range(len(week)):

        if state == week[i]:

            return i

def change\_default(state):

    # 转换default

    default = ["no","yes","unknown"]

    for i in range(len(default)):

        if state == default[i]:

            return i

def read(filename):

    bank = pd.read\_csv(filename, sep=';')

    data = [[] for i in range(len(bank))]

    for i in range(len(data)):

        data[i].append(bank['age'][i])

        data[i].append(change\_job(bank['job'][i]))

        data[i].append(change\_marital(bank['marital'][i]))

        data[i].append(change\_education(bank['education'][i]))

        data[i].append(change\_default(bank['default'][i]))

        data[i].append(change\_default(bank['housing'][i]))

        data[i].append(change\_default(bank['loan'][i]))

        data[i].append(change\_contact(bank['contact'][i]))

        data[i].append(change\_month(bank['month'][i]))

        data[i].append(change\_week(bank['day\_of\_week'][i]))

        data[i].append(bank['duration'][i])

        data[i].append(bank['campaign'][i])

        data[i].append(bank['pdays'][i])

        data[i].append(bank['previous'][i])

        data[i].append(change\_poutcome(bank['poutcome'][i]))

        data[i].append(bank['emp.var.rate'][i])

        data[i].append(bank['cons.price.idx'][i])

        data[i].append(bank['cons.conf.idx'][i])

        data[i].append(bank['euribor3m'][i])

        data[i].append(bank['nr.employed'][i])

        data[i].append(change\_binary(bank['y'][i]))

    for x in range(len(data)):

        for y in range(len(data[x])):

            # print(data[x][y])

            data[x][y] = float(data[x][y])

    return data

def euclideanDistance(p1, p2,len):

    #计算欧式距离

    dis=0

    for x in range(len):

        dis += pow((p1[x] -  p2[x]), 2)

    return math.sqrt(dis)

if \_\_name\_\_ == '\_\_main\_\_':

    filename = r"E:\\workworkwork\\GitHub\WinnieLearning\\python\\分类算法\\knn分类\\bank-additional.csv"

    data = read(filename)

    train\_data=[]

    test\_data=[]

    #拆分数据集

    for id in range(len(data)):

        if random.random() <= 0.7:

            train\_data.append(data[id])

        else:

            test\_data.append(data[id])

    print("划分的训练集数:", len(train\_data), "划分的测试集数:", len(test\_data))

    predict=[]

    k = 7

    for x in range(len(test\_data)):

        #选取距离最近的k个实例 test\_data(需要分类的测试集点)

        dis=[]

        length = len(test\_data[x]) - 1

        for id in range(len(train\_data)):

            dis.append((train\_data[id], euclideanDistance(train\_data[id],test\_data[x],length)))

        dis.sort(key = operator.itemgetter(1))

        neighbors=[]

        for id in range(k):

            neighbors.append(dis[x][0])

        #获取距离最近的k个实例中占比例较大的分类

        classvotes = {}

        for x in range(len(neighbors)):

            response=neighbors[x][-1]

            if response in classvotes:

                classvotes[response]+=1

            else:

                classvotes[response]=1

            sortedVotes = sorted(classvotes.items(), key=operator.itemgetter(1), reverse=True)

        res = sortedVotes[0][0]

        predict.append(res)

    # 计算准确率

    accuracy = 0

    for x in range(len(test\_data)):

        if test\_data[x][-1]==predict[x]:

            accuracy += 1

    accuracy = accuracy / float(len(test\_data)) \* 100

    print('准确率：', accuracy)

    # 可视化

    X = test\_data

    X1 = [n[0] for n in X]

    #X2 = [n[8] for n in X]   # 以month划分

    #X2 = [n[10] for n in X]  # 以duration划分

    X2 = [n[5] for n in X]  # 以housing划分

    X3 = [n[-1] for n in X]

    # 画图

    plt.xlabel("age")

    #plt.ylabel("month")

    #plt.ylabel("duration")

    plt.ylabel("housing")

    plt.scatter(X1, X2, c = X3, cmap = plt.cm.spring, edgecolor = 'k')

    plt.show()