南京理工大学泰州科技学院
实 验 报 告 书

课程名称：《大数据可视化》

实验题目：实验一

Iris 数据集的数据分析

实现 Vector 类

班　　级：20 信管

学　　号：2009120112

姓　　名：陆姝婷

指导教师：叶志鹏

成　　绩：

批阅教师：叶志鹏

年　　月　　日

一、实验目的

1. 回顾 Python 基础知识。
2. 掌握 Python 数据读写
3. 掌握 Python 面向对象的概念。
4. 掌握 Python 模块与包的概念

二、实验设备

1. PyCharm
2. random 、 numpy 、 math 库。

三、实验内容

3.1 Iris 数据集的数据分析

- **读取 iris.data 数据集到内存中，并存储为列表命名为 iris_list，里面的元素可以是自定义对象，也可以是 2 维列表，并输出 iris.data 有多少个样本个数，以及有多少种类别。**

① 通过 open() 方法打开文件

② 用 File 的方法 readlines() 读取所有行并返回列表

③ 通过 len() 方法获取列表长度，即数据的样本数

④ 创建 iris_list 空列表，用 for 循环遍历一开始得到的列表。通过 append() 方法添加，并在 append() 里嵌套 split 对每一串数据进行按逗号切片

⑤ 创建一个 list 空列表，先通过 append() 方法添加第一行第五列的种类【我通过 replace() 方法将末尾的转义字符变为空，可变可不变结果都一样】，并且将记录种类的计数器 count 起始值定位 1

⑥ 通过自定义函数 fun1(list, data) 来循环遍历比较每个样本的第五列看是否已经在 list 列表中，不在 list 中的就添加进去，并且返回 True ，否则返回 False ，通过调用自定义函数，是真就 count + 1 ,遍历完成最后获得类别个数

```
count = 1
list = []
iris_list = []
for i in range(iris_length):
    iris_list.append(iris_data[i].split(","))
print(iris_list)
list.append(iris_list[0][4].replace('\n', ''))


def fun1(list, data):
    for i in range(len(list)):
        if list[i] == data:
            return False
    list.append(data)
    return True


for i in range(iris_length):
    if fun1(list, iris_list[i][4].replace('\n', '')):
        count = count + 1

print(list)
print(f"iris.data中有类别: {count}")
```

- **按照第一列属性（sepal length 花萼长度）将上步操作得到的列表升序排序并打印结果。**

  ① 通过调用 sort() 方法，实现排序

  ② 内部参数具体为按照第一列的值进行排序所以需要使用 key = (lambda x:float(x[0]) 用来设置排序按照什么来，并且通过强制类型转换将字符串转为浮点数

  ③ 并且将 reverse 的值设置为 False ，也就是升序

- **实现一个 Python 函数，能够实现对 iris_list 的有放回随机抽样，函数参数为抽样列表 data，抽样个数 number ，并测试打印结果。**

1

2

```
def sampling_with_replacement(data , number) :
# todo
```

① 通过调用自定义方法 sampling_with_replacement(data,number) 实现

② 方法形参 data 是抽样的样本， number 是抽样的个数

③ 方法内循环抽样，通过 random 库的 choice() 进行有放回抽样

④ choice(序列) 方法的作用：从非空序列中随机选取一个数据并返回，该序列可以是 list、tuple、str、set，且每次选取都不会影响原序列，每一次选取都是基于原序列

- **实现一个 Python 函数，能够实现对 iris_list 的无放回随机抽样，函数参数为抽样列表 data，抽样个数 number ，并测试打印结果。**

```
1  def sampling_without_replacement(data , number) :
2  # todo
```

① 通过调用自定义函数 sampling_without_replacement(data,number) 实现

② 方法形参 data 是抽样的样本， number 是抽样的个数

③ 通过 random 库的 sample() 进行无放回抽样

④ sample(list, k) 返回一个长度为 k 新列表，新列表存放从 list 中所产生的 k 个随机唯一的元素

- **统计 iris 各列属性均值，方差，标准差，中位数并打印输出。**

① 通过定义一个自定义函数 avg(data, line) 来遍历当前列的值进行累加之后返回累加和除样本数的值，即此列的均值

② 形参分别传入的是列表和列数，在进行累加时还需要通过强制类型转换 float() 将字符串转为浮点数，且返回默认保留六位小数的均值，输出采用占位符 % 形式输出

③ 通过调用 numpy 库的 var() 进行方差计算

④ 首先创建一个新列表 iris_list_new 用来存储遍历所有原始数据去掉最后一列并且每个数据都转为浮点数

⑤ var（a，axis=None，dtype=None，out=None，ddof=0，keepdims=<no value>， *， where=<no value>) 此处主要用到前三个参数，主要为：进行方差计算的数据； axis = 1 表示按行计算方差，axis = 0 表示按列计算方差，此处需要用到第二个；用于计算方差的类型，此处选择 np.float64 ，因为对于浮点数输入可能会导致 结果不准确，特别是对于 float32。使用关键字指定更高精度的累加器可以缓解此问题

⑥ 通过调用 numpy 库的 std() 进行标准差计算

⑦ std (a, axis=None, dtype=None, out=None, ddof=0) 此处主要用到前两个参数：a：需计算标准差的数组； axis： int, 可选，计算标准差的轴，此处写 0 为按列计算

⑧ 通过调用 numpy 库的 percentile() 进行中位数计算

⑨ percentile(a, q, axis=None, out=None, overwrite_input=False, interpolation='linear', keepdims=False) 此处只需用到前三个参数即可，主要为：a:输入数组；q：要计算的百分位数，取值在 0~100 之间；axis: 0 为按列计算，1 为按行计算

⑩ 主要是 q 的取值，此处要算中位数，所以取 50（例如：假设某个考生在入学考试中的语文的原始分数为 54 分。相对于参加同一考试的其他学生来说，他的成绩如何并不容易知道。但是如果分数 54 分恰好对应的是第 70 百分位数，我们就能知道大约 70%的学生的考分比他低，而约 30%的学生考分比他高。这里的 q = 70）

```
def avg(data, line):
    sum = 0
    for i in range(len(data)):
        sum += float(data[i][line - 1])
    return sum / iris_length


line1 = avg(iris_list, 1)
print("第一列的均值: %f" % line1)
line2 = avg(iris_list, 2)
print("第一列的均值: %f" % line2)
line3 = avg(iris_list, 3)
print("第一列的均值: %f" % line3)
line4 = avg(iris_list, 4)
print("第一列的均值: %f" % line4)

print("=====================================================")

iris_list_new = []
for i in range(iris_length):
    iris_list_new.append(iris_list[i][0:-1])
    iris_list_new[i] = [float(x) for x in iris_list_new[i]]

np_var1 = np.var(iris_list_new, axis=0, dtype=np.float64)
print(f"前四列的方差: {np_var1}")

print("=====================================================")

np_std = np.std(iris_list_new, axis=0)
print(f"前四列的标准差: {np_std}")
```

3.2 实现 Vector 类

- **实现 Vector 类，并完成 __init__(self, data) 方法。**

```
1
2
3
class Vector：def __init__( self，
    data)：
    # todo
```

① 创建一个名为 Vector 的类

② 创建有参构造方法 __init__(self, data) ，进行初始化

- **实现 __len__(self) 特殊方法，能够通过 len(vector) 获取到向量的维度。**

① 创建一个名为 __len__(self) 的方法

② 通过调用 numpy 库的 array() 传递 Python 的序列对象创建数组，如果传递的是多层嵌套的序列，将创建多维数组

③ 数组的大小可以通过其 shape 属性获得，当数组或矩阵是一维时，只能使用 shape[0]，返回的是数组或矩阵中元素的个数，此时返回的就是向量的维度

- **实现 __str__(self) 特殊方法，能够通过 print(vector) 获取到向量的元素。**

① 创建一个名为 __str__(self) 的类

② 通过创建的实例调用该方法， self.x 和 self.y 分别获得该实例对应的元素并通过 format() 函数进行格式化打印

- **实现向量的加法运算，如 vec3 = vec1 + vec2 。**

① 创建一个名为 __add__(self, other) 的方法

② 参数分别传入调用此方法的实例对象和与此相加的另一个实例对象

③ 在方法中分别调用相关的属性进行相加

④ 最后调用函数的有参构造，进行新向量的实例化并返回给调用处

- **实现向量的内积运算，** $scale = \vec{x_1} * \vec{x_2} = x_1^T * x_2$ 。

① 创建一个名为 __scale__(self, other) 的方法

② 参数分别传入调用此方法的实例对象和与此相加的另一个实例对象

③ 在方法中分别调用相关的属性进行相乘

④ 最后调用函数的有参构造，进行新向量的实例化并返回给调用处

```python
import numpy as np
import math


class Vector(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __len__(self):
        a = np.array((self.x, self.y))
        return a.shape[0]

    def __add__(self, other):
        x = self.x + other.x
        y = self.y + other.y
        return Vector(x, y)

    def __scale__(self, other):
        x = self.x * other.x
        y = self.y * other.y
        return Vector(x, y)

    def __abs__(self):
        return math.sqrt(self.x ** 2 + self.y ** 2)

    def __str__(self):
        return "Vector({}, {})".format(self.x, self.y)


def main():
```

Vector › __abs__()