**课后作业二：概率分类法**

任务：使用贝叶斯估计或MLE（最大似然估计），来预测鸢尾花数据集中花的种类。

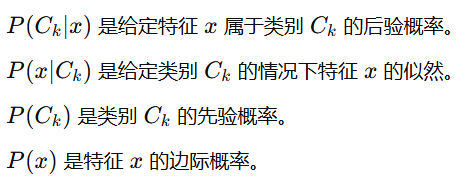
数据集：鸢尾花数据集是统计学和机器学习中用于分类的经典数据集。该数据集包含了三种不同的鸢尾花：Setosa、Versicolor和Virginica，每种各50个样本。每个样本有四个属性：萼片长度、萼片宽度、花瓣长度和花瓣宽度，所有的测量单位都是厘米。数据集根据4:1的比例划分为训练集和测试集。概率分类法是一种基于概率理论的方法，适合处理此类分类问题。

**原理：**

贝叶斯定理公式：



其中：



**实现：**

要实现通过贝叶斯分类进行预测，需要完成以下几步

·计算每个类别的先验概率。

·对每个类别和每个特征，计算其均值和标准差。

·使用高斯分布计算给定特征的似然。

·使用贝叶斯定理计算后验概率。

·选择具有最高后验概率的类别作为预测类别。

1.导入必要的库

from sklearn.model\_selection import train\_test\_split #机器学习中非常重要的库，包括一些分类、回归、聚类、降维、模型选择和预处理

from collections import defaultdict

from math import sqrt, pi, exp

import pandas as pd #用来分析结构化数据

import numpy as np #提供高性能的矩阵运算

import csv #读写文件的库

2.导入训练数据并提取特征值和目标值

"sepal length (cm)", "sepal width (cm)", "petal length (cm)", "petal width (cm)"为特征值，"species"为目标值

iris\_data = pd.read\_csv(r'D:\dataenclorse\second\iris\_train.csv')

X = iris\_data[["sepal length (cm)", "sepal width (cm)", "petal length (cm)", "petal width (cm)"]].values

y = iris\_data["species"].values

3.划分数据

# 构造训练数据和测试数据

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

4.构造贝叶斯分类模型并训练该模型

（1）定义计算先验概率的函数

# 计算每个类别的先验概率

def calculate\_prior(y):

    class\_counts = defaultdict(int)

    for label in y:

        class\_counts[label] += 1

    total\_count = len(y)

    priors = {label: count / total\_count for label, count in class\_counts.items()}

    return priors

（2）手动实现计算特征值和标准差的函数

# 计算每个类别和每个特征的均值和标准差

def calculate\_mean\_std(X, y):

    separated = defaultdict(list)

    for i in range(len(y)):

        separated[y[i]].append(X[i])

    summary = {}

    for label, instances in separated.items():

        summary[label] = [(np.mean(attribute), np.std(attribute)) for attribute in zip(\*instances)]

    return summary

（3）实现高斯分布的概率密度函数

def gaussian\_probability(x, mean, std):

    exponent = exp(-((x - mean) \*\* 2 / (2 \* std \*\* 2)))

    return (1 / (sqrt(2 \* pi) \* std)) \* exponent

（4）实现计算特征的似然的函数

# 计算给定特征的似然

def calculate\_likelihood(summary, x):

    likelihoods = {}

    for label, stats in summary.items():

        likelihood = 1

        for i in range(len(stats)):

            mean, std = stats[i]

            likelihood \*= gaussian\_probability(x[i], mean, std)

        likelihoods[label] = likelihood

return likelihoods

（5）使用贝叶斯定理计算后验概率

def calculate\_posterior(priors, likelihoods):

    posteriors = {}

    for label in priors:

        posteriors[label] = priors[label] \* likelihoods[label]

    total\_posterior = sum(posteriors.values())

    for label in posteriors:

        posteriors[label] /= total\_posterior

    return posteriors

1. 手动实现的朴素贝叶斯分类器预测函数

def naive\_bayes\_predict(X\_train, y\_train, X\_new):

    priors = calculate\_prior(y\_train)

    summary = calculate\_mean\_std(X\_train, y\_train)

    predictions = []

    for x in X\_new:

        likelihoods = calculate\_likelihood(summary, x)

        posteriors = calculate\_posterior(priors, likelihoods)

        best\_label = max(posteriors, key=posteriors.get)

        predictions.append(best\_label)

    return predictions

5.加载iris\_test.csv的数据并对iris\_test.csv进行预测

# 读取待预测的新数据点

iris\_test\_data = pd.read\_csv(r'D:\dataenclorse\second\iris\_test.csv')

X\_new = iris\_test\_data[["sepal length (cm)", "sepal width (cm)", "petal length (cm)", "petal width (cm)"]].values

# 预测新数据点的类别

predictions = naive\_bayes\_predict(X\_train, y\_train, X\_new)

print("预测的目标类别是：{}".format(predictions))

6.预测结果与加载的数据一起保存到test.csv文件中

getdata和getdata2函数与作业一中一模一样，仅仅有读取文件的内容不同

file\_path = r'D:\dataenclorse\second\test\_manual\_bayes.csv'

with open(file\_path, 'w', encoding='utf-8', newline='') as f:

    fieldnames = ['sepal length(cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)', 'class']

    f\_csv = csv.DictWriter(f, fieldnames=fieldnames)

    f\_csv.writeheader()

    for i in range(len(predictions)):

        f\_csv.writerow({

            'sepal length(cm)': X\_new[i][0],

            'sepal width (cm)': X\_new[i][1],

            'petal length (cm)': X\_new[i][2],

            'petal width (cm)': X\_new[i][3],

            'class': predictions[i]

        })

print(f"预测结果已保存到 {file\_path}")

1. 结果如下：

