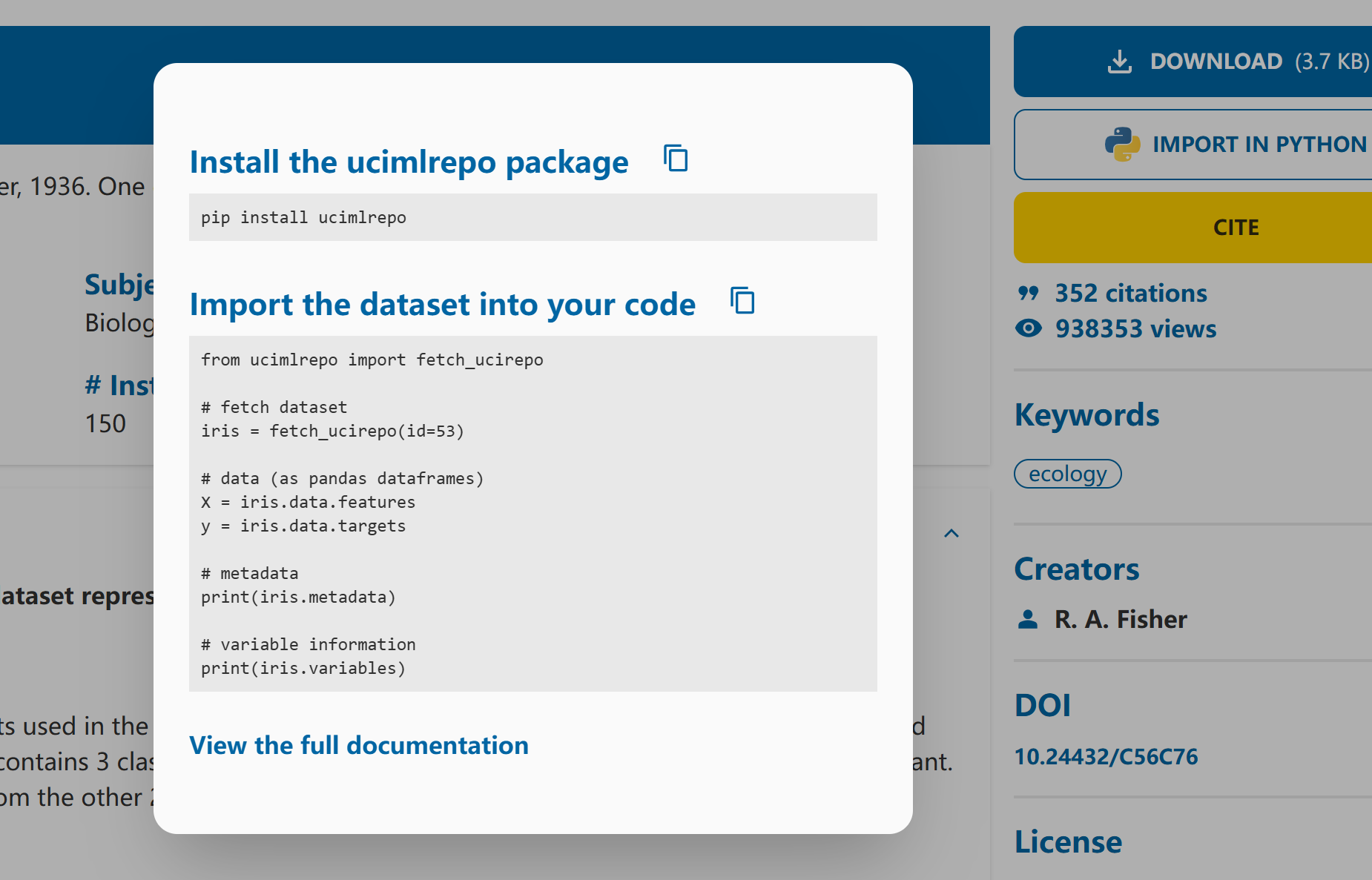
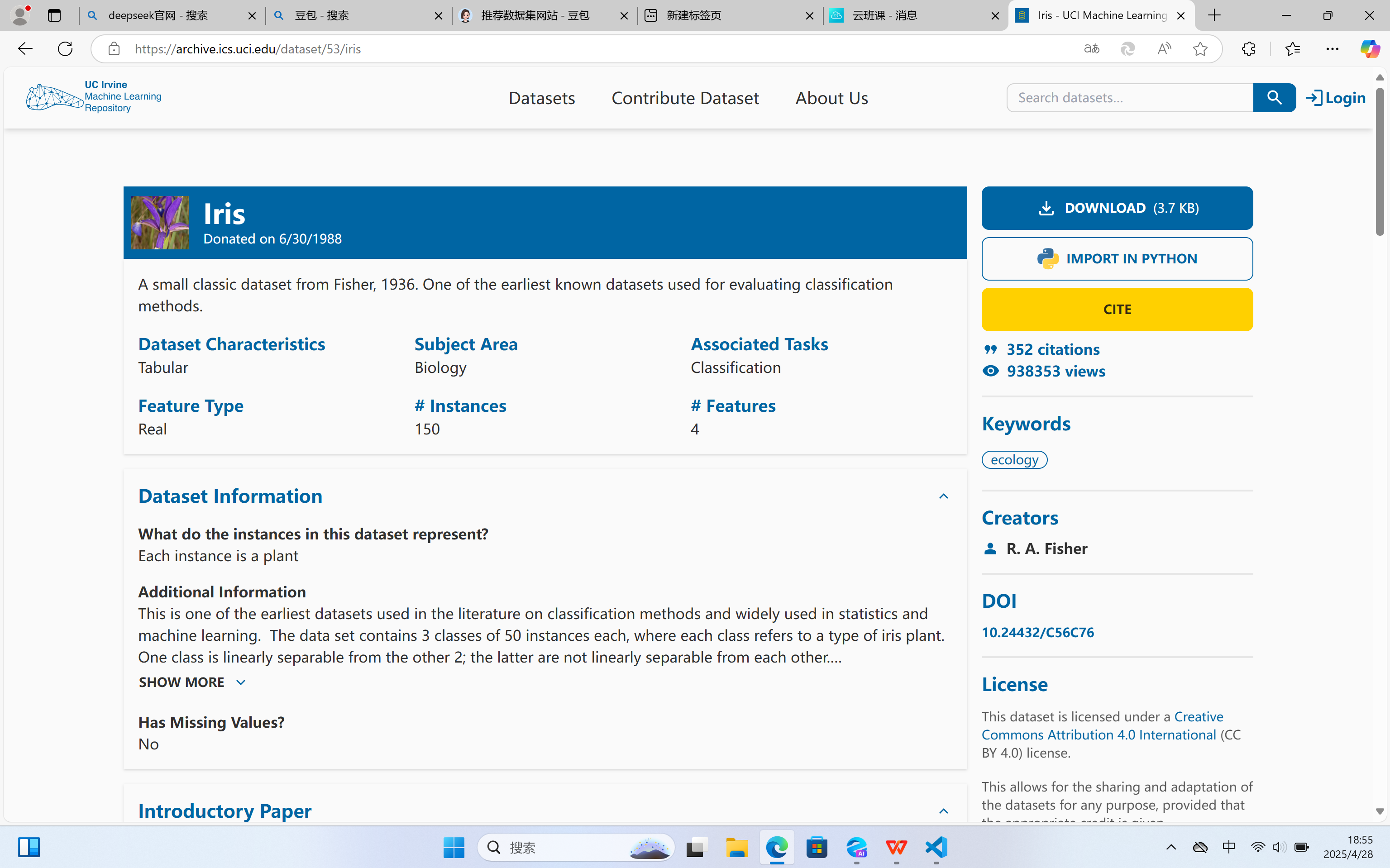
# 鸢尾花数据集数据分析报告

## 一、基本信息

班级：人资243  
姓名：张涵宇  
学号：2024311879

任务完成情况：独立完成数据采集、代码编写、分析及报告撰写的全部工作。

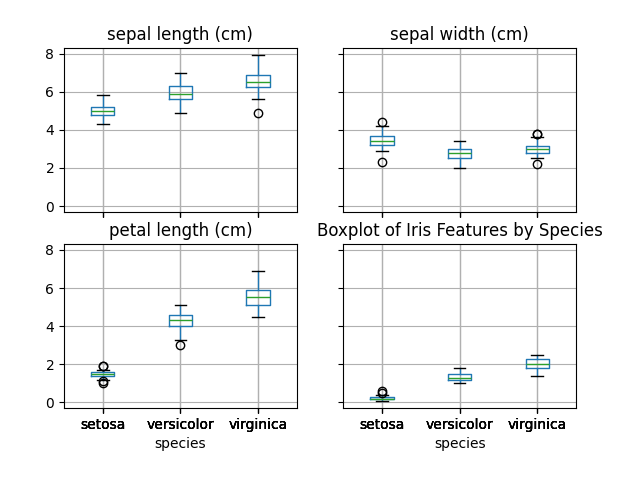
## 数据采集



## 三、统计分析方法应用

### （一）描述性统计分析

1.使用Pandas库对鸢尾花数据集进行描述性统计分析。对 4 个数值型特征（花萼长度、花萼宽度、花瓣长度、花瓣宽度）计算均值、中位数、标准差、最小值和最大值。（因为没有缺失值，不用对数据预处理）



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | length | width | length | width |
| count | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| mean | 5.843333 | 3.057333 | 3.758000 | 1.199333 |
| std | 0.828066 | 0.435866 | 1.765298 | 0.762238 |
| min | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| 25% | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| 50% | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| 75% | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| max | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

2.结果显示，样本完整；花萼长度的均值5.843333cm，花萼宽度均值3.057333cm，花瓣长度均值3.758000cm，花瓣宽度均值1.199333cm；花萼长度的标准差是0.828066cm，花萼宽度标准差0.435866cm，花瓣长度标准差1.765298cm，花瓣宽度标准差0.762238cm，这里花瓣长度的标准差最大，表明花瓣长度的样本波动相对较大，花萼宽度的标准差最小，说明花萼宽度的数据相对较为集中；花萼长度最小值为 4.3cm，花萼宽度最小值2cm，花瓣长度最小值1cm，花瓣宽度最小值0.1cm......<见表格>

3.代码如下：

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

# 加载鸢尾花数据集

iris = load\_iris()

# 将数据转换为 DataFrame 格式

iris\_df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

# 添加品种列

iris\_df['species'] = iris.target

species\_mapping = {0: iris.target\_names[0], 1: iris.target\_names[1], 2: iris.target\_names[2]}

iris\_df['species'] = iris\_df['species'].map(species\_mapping)

# 计算描述性统计量

stats = iris\_df[['sepal length (cm)', 'sepal width (cm)',

                 'petal length (cm)', 'petal width (cm)']].describe()

print(stats)

# 绘制箱线图

iris\_df.boxplot(column=['sepal length (cm)', 'sepal width (cm)',

                        'petal length (cm)', 'petal width (cm)'], by='species')

plt.title('Boxplot of Iris Features by Species')

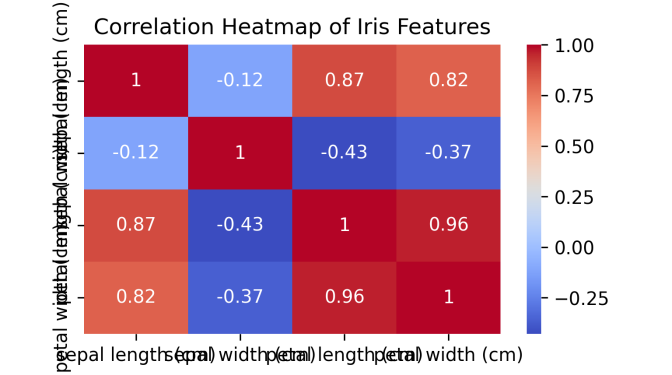
plt.suptitle('')

plt.show()

### （二）相关性分析

1.利用 Pandas 的corr()函数构建相关系数矩阵。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ----- | Sepal-length | sepal-width | petal-length | petal-width |
| Sepal-length | 1.000000 | -0.117570 | 0.871754 | 0.817941 |
| Sepal-width | -0.117570 | 1.000000 | -0.428440 | -0.366126 |
| Petal-length | 0.871754 | -0.428440 | 1.000000 | 0.962865 |
| Petal-width | 0.817941 | -0.366126 | 0.962865 | 1.000000 |



2.结果表明，

花瓣长度与花瓣宽度：相关系数高达 0.96 ，二者关联紧密。

花萼长度与花瓣长度：相关系数为 0.87 ，表明二者存在较高程度的正相关关系。

花萼长度与花瓣宽度：相关系数 0.82，说明二者也有较强正相关。

3.代码如下：

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

# 加载鸢尾花数据集

iris = load\_iris()

# 将数据转换为 DataFrame 格式

iris\_df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

# 计算皮尔逊相关系数矩阵

correlation\_matrix = iris\_df.corr()

print(correlation\_matrix)

# 设置图片清晰度

plt.rcParams['figure.dpi'] = 300

# 绘制热力图

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

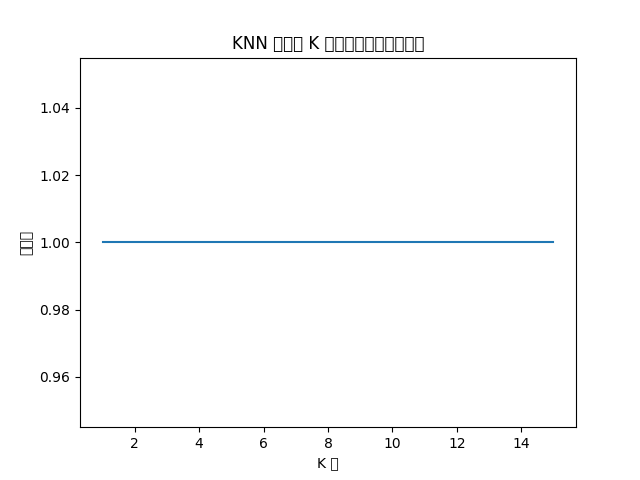
plt.title('Correlation Heatmap of Iris Features')

plt.show()

## 四、机器学习算法应用

### （一）K 近邻（KNN）分类算法

1.使用Scikit-learn库实现 KNN 算法。选取 K 值为2进行模型训练和预测。



2.结果表明：当 K 值为 2 时，测试集上的准确率为: 1.00，说明这个KNN分类算法在这个模型上十分好用

3.优缺点分析：

优点：数据训练简单，算法成熟。

缺点：大量数据，计算复杂的数据KNN算法效果不好

4.代码如下：

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

# 加载鸢尾花数据集

iris = load\_iris()

X = iris.data

y = iris.target

# 划分训练集和测试集，比例为7:3

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=0.3, random\_state=42)

# 选取K值为2，训练并预测

k = 2

knn = KNeighborsClassifier(n\_neighbors=k)

knn.fit(X\_train, y\_train)

y\_pred = knn.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"当K值为{k}时，测试集上的准确率为: {accuracy:.2f}")

# 探究K值对模型性能的影响

k\_values = np.arange(1, 16)

accuracies = []

for k in k\_values:

    knn = KNeighborsClassifier(n\_neighbors=k)

    knn.fit(X\_train, y\_train)

    y\_pred = knn.predict(X\_test)

    accuracy = accuracy\_score(y\_test, y\_pred)

    accuracies.append(accuracy)

# 绘制准确率随K值变化的折线图

plt.plot(k\_values, accuracies)

plt.xlabel('K 值')

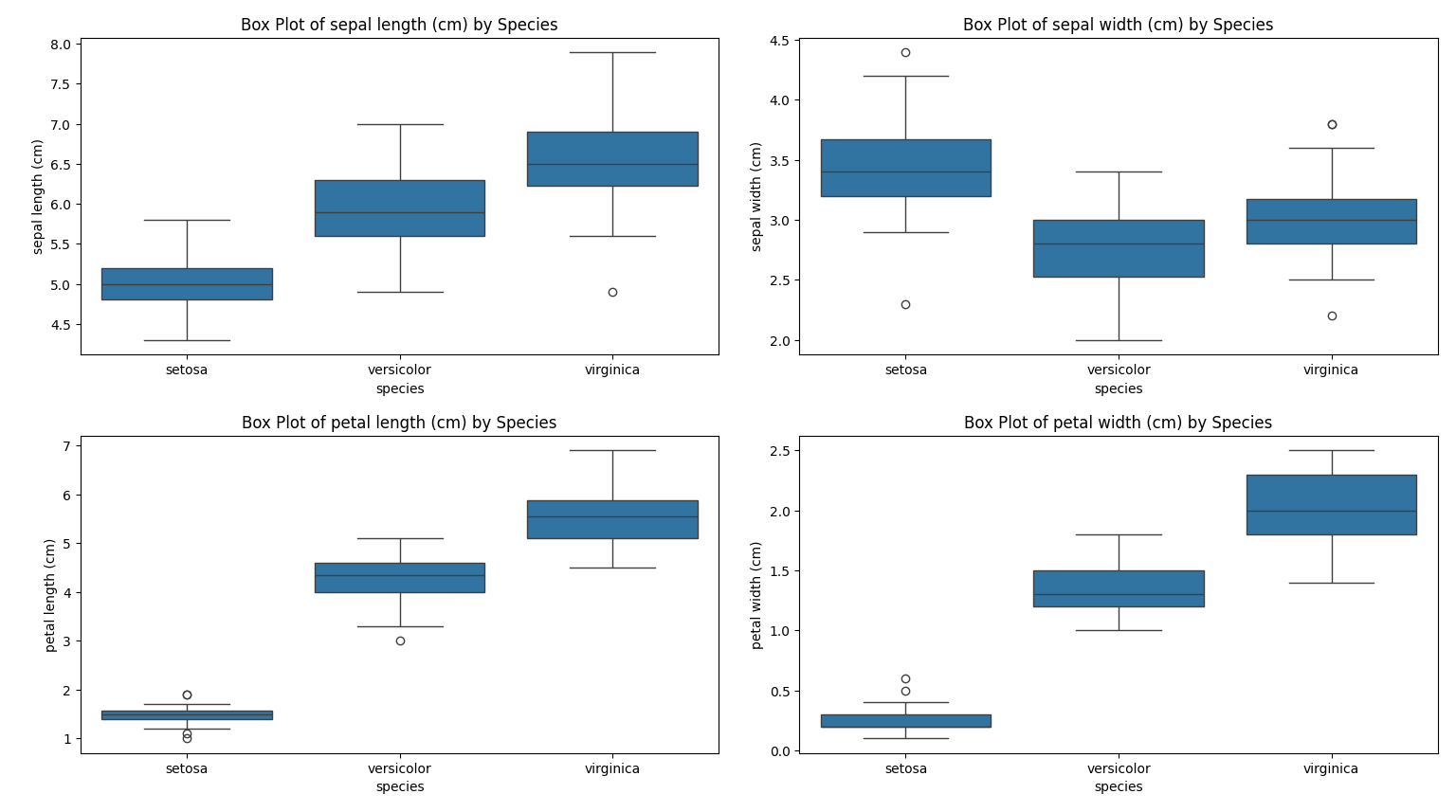
plt.ylabel('准确率')

plt.title('KNN 算法中 K 值对模型准确率的影响')

plt.show()

### **（三）**逻辑回归，决策树，PCA分析算法介绍

### Figure_3Figure_1



1.算法介绍（第一次用）

逻辑回归：训练逻辑回归模型并计算准确率。通过创建网格点，预测每个网格点的类别，进而绘制决策边界。

决策树：训练决策树模型并计算准确率。同样使用网格点预测类别，绘制决策边界。

主成分分析：将鸢尾花数据集的 4 维特征降维到 2 维，绘制散点图展示不同类别在降维空间中的分布。

可视化：使用matplotlib将逻辑回归和决策树的决策边界以及 PCA 降维结果绘制在同一图片上。

2.结果如下：

逻辑回归模型的准确率: 0.82

决策树模型的准确率: 0.67

3.代码如下：

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.decomposition import PCA

from sklearn.metrics import accuracy\_score

# 加载鸢尾花数据集

iris = load\_iris()

X = iris.data[:, :2]

y = iris.target

# 划分训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=0.3, random\_state=42)

# 逻辑回归分析及可视化

logreg = LogisticRegression(max\_iter=1000)

logreg.fit(X\_train, y\_train)

y\_pred\_logreg = logreg.predict(X\_test)

accuracy\_logreg = accuracy\_score(y\_test, y\_pred\_logreg)

print(f"逻辑回归模型的准确率: {accuracy\_logreg:.2f}")

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.02),

                     np.arange(y\_min, y\_max, 0.02))

Z = logreg.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)

plt.contourf(xx, yy, Z, alpha=0.4)

plt.scatter(X[:, 0], X[:, 1], c=y, alpha=0.8)

plt.title('逻辑回归决策边界')

plt.xlabel('Sepal length')

plt.ylabel('Sepal width')

# 决策树分析及可视化

dtree = DecisionTreeClassifier(random\_state=42)

dtree.fit(X\_train, y\_train)

y\_pred\_dtree = dtree.predict(X\_test)

accuracy\_dtree = accuracy\_score(y\_test, y\_pred\_dtree)

print(f"决策树模型的准确率: {accuracy\_dtree:.2f}")

Z = dtree.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.subplot(1, 3, 2)

plt.contourf(xx, yy, Z, alpha=0.4)

plt.scatter(X[:, 0], X[:, 1], c=y, alpha=0.8)

plt.title('决策树决策边界')

plt.xlabel('Sepal length')

plt.ylabel('Sepal width')

# 主成分分析（PCA）及可视化

X = iris.data

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X)

plt.subplot(1, 3, 3)

plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=y, cmap='viridis')

plt.xlabel('第一主成分')

plt.ylabel('第二主成分')

plt.title('鸢尾花数据集的 PCA 降维可视化')

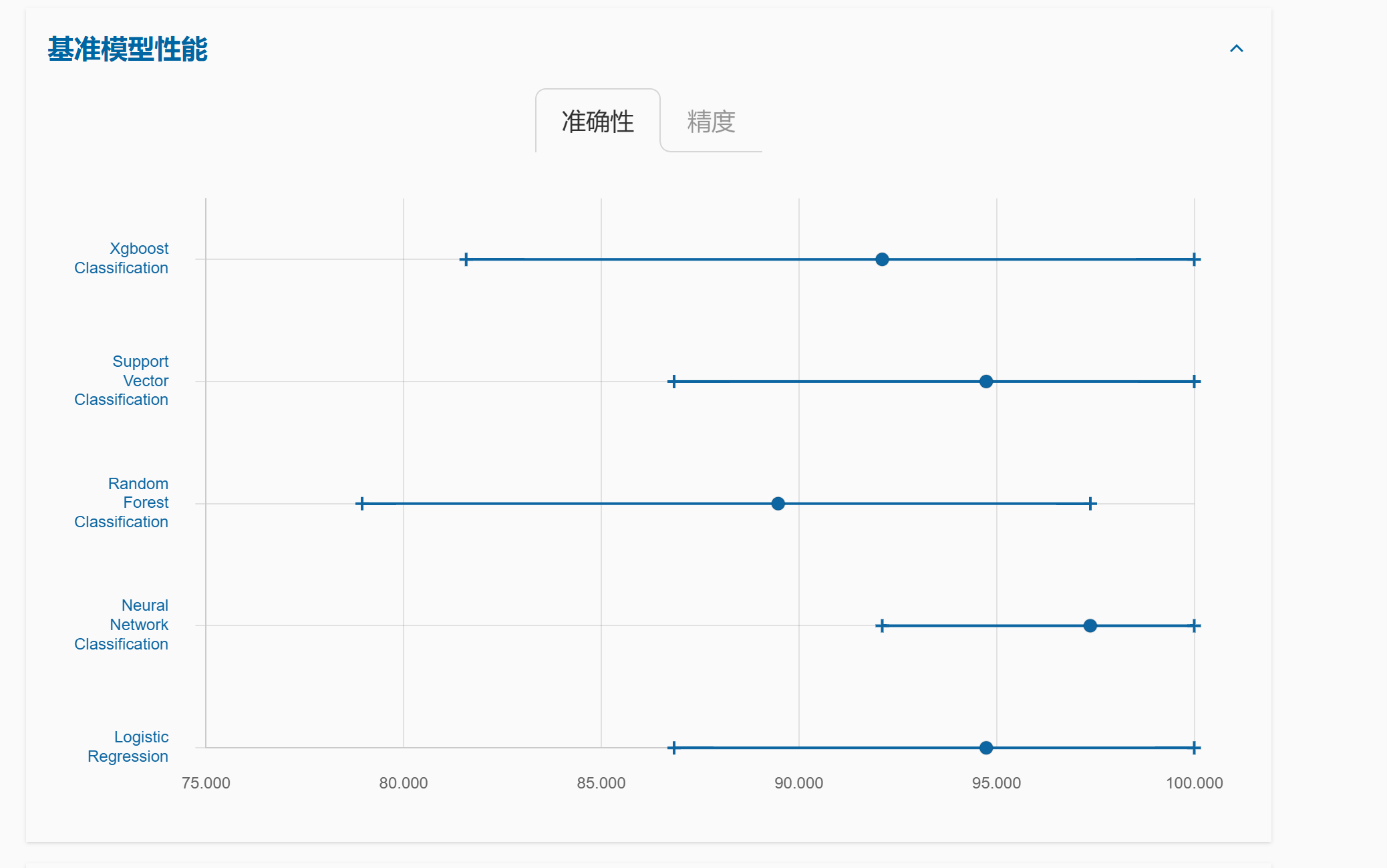
plt.colorbar(label='鸢尾花类别')

plt.tight\_layout()

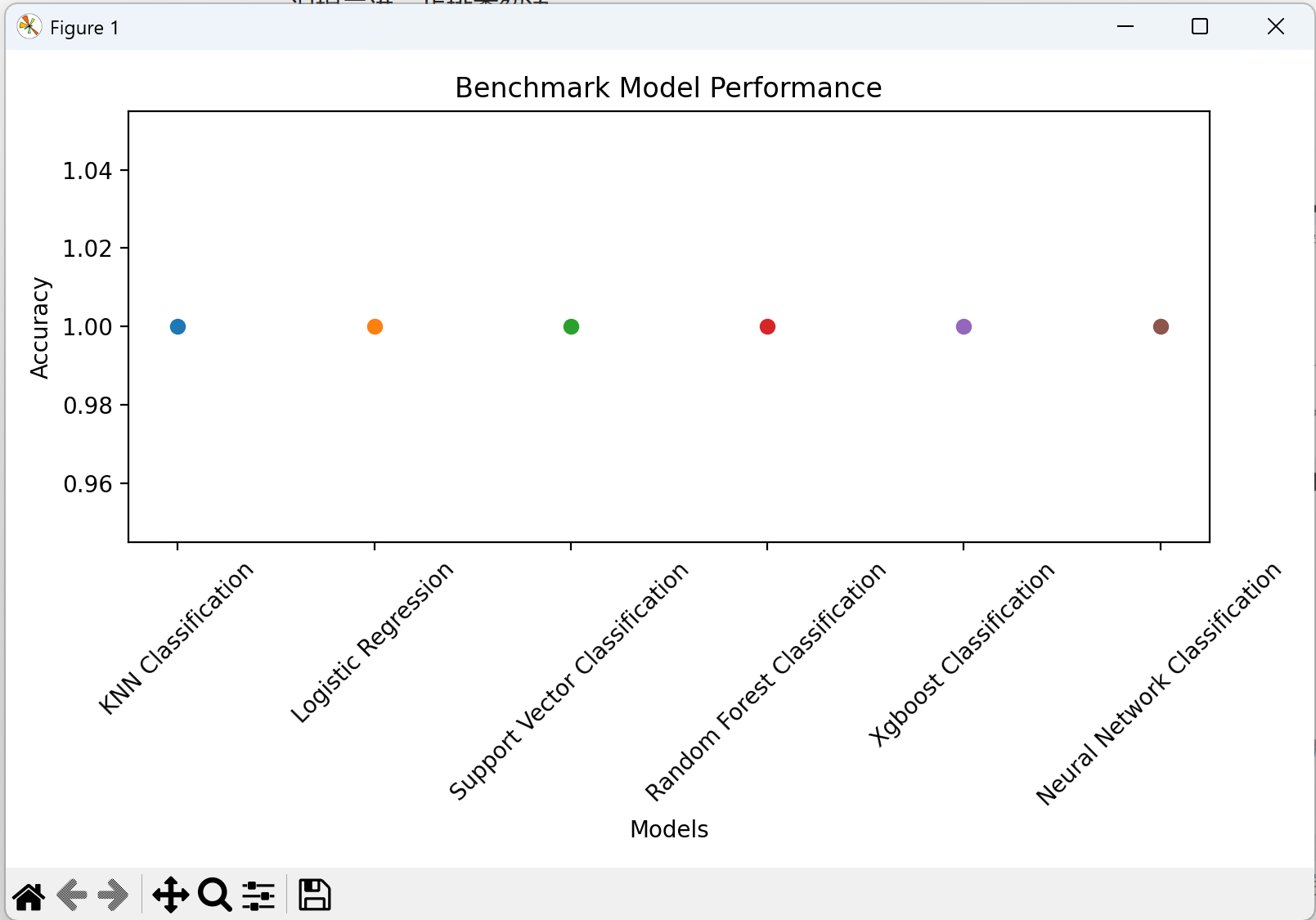
plt.show()

## 结论

在Iris模型上，使用KNN分类算法是最好的方法，在官网上，我们可以看到



我去测试了一下，结果如下：



不管是哪个方法，都是100%的准确率，说明这个数据集这几个方法都可以，并且，数据集太小了，还分不出哪个模型更好。