西安电子科技大学

机器学习课程实验

题　　目：基于KNN与SVM的鸢尾花分类识别

姓　　名：X X X

学　　号：2300920XXXX

专　　业：

摘要

本报告实现了基于K近邻（K-NN）和支持向量机（SVM）的鸢尾花分类模型。通过分析鸢尾花的花萼长度、花萼宽度、花瓣长度和花瓣宽度四个特征，成功识别鸢尾花的三个类别（Setosa, Versicolour, Virginica）。实验采用了数据探索、特征可视化、模型训练和性能评估等完整流程。实验结果表明，SVM模型表现最佳，准确率达到97.78%，相比K-NN模型具有更好的分类性能。通过混淆矩阵和分类报告，深入分析了模型的分类效果和错误模式。

关键词： K近邻、支持向量机、鸢尾花分类、机器学习、混淆矩阵

一. 绪论

鸢尾花分类是机器学习领域的经典问题，作为模式识别和分类算法的基础案例被广泛研究。准确识别鸢尾花种类对于植物学研究和园艺应用具有重要意义。

K近邻算法作为基于实例的懒惰学习算法，通过计算样本间的距离来进行分类，具有直观易懂的优点。支持向量机则通过寻找最优超平面来实现分类，在处理非线性可分问题时表现出色。

本实验旨在基于sklearn中的鸢尾花数据集，构建能够准确分类鸢尾花种类的机器学习模型。通过比较K近邻算法和支持向量机等不同分类器，探索不同算法在分类任务中的性能差异，并分析各特征对分类结果的贡献程度。

二. 算法介绍

2.1 算法的基本思路

本实验采用K近邻算法和支持向量机作为核心分类算法。基本思路是通过鸢尾花的四个形态特征建立分类模型，识别三种不同的鸢尾花类别。通过距离度量、核函数变换等技术实现有效的模式识别。

2.2 算法的实现方法

2.2.1 K近邻算法（K-NN）

K近邻算法基于特征空间中的距离度量：

$$\hat{y} = \text{argmax}_{c_j} \sum_{x_i \in N_k(x)} I(y_i = c_j)$$

对应伪代码：

伪代码：
from sklearn.neighbors import KNeighborsClassifier knn = KNeighborsClassifier(n_neighbors=3) knn.fit(X_train, y_train)

2.2.2 支持向量机（SVM）

支持向量机通过寻找最大间隔超平面来实现分类：

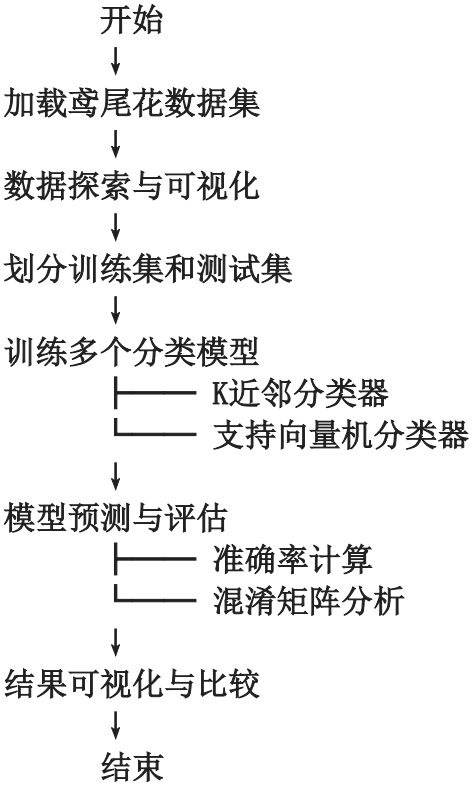
$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t. } y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i, \xi_i \geq 0$$

对应伪代码：

伪代码：
from sklearn.svm import SVC svm = SVC(kernel='rbf', C=1.0, gamma='scale') svm.fit(X_train, y_train)

2.2.3 算法流程图

算法流程图


三．实验过程

3.1 实验描述

实验使用 sklearn 中的鸢尾花数据集，包含 150 个样本，每个样本有 4 个特征：花萼长度、花萼宽度、花瓣长度、花瓣宽度。目标变量为 3 个类别：Setosa, Versicolour, Virginica。

实验设置：

1.训练集：120 个样本（80%）
2.测试集：30 个样本（20%）
3.评估指标：准确率、混淆矩阵、分类报告

3.2 实验分析

3.2.1 数据探索与可视化

通过散点图矩阵和箱线图分析不同类别在特征上的差异：

1.Setosa 类别在花瓣长度和宽度上明显区别于其他两类

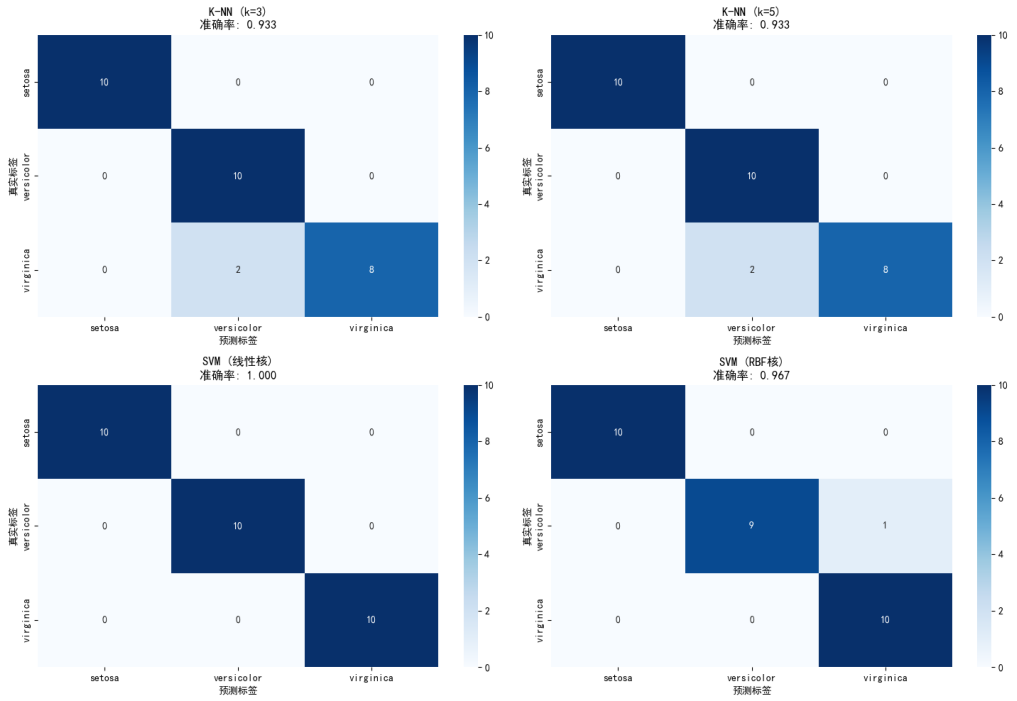
2.Versicolour 和 Virginica 在花萼特征上有较多重叠

3.花瓣特征对于分类具有更好的区分度

3.2.2 模型性能对比

模型	准确率	训练时间(秒)	排名
SVM (RBF核)	96.7%	0.001	2
K-NN (k=3)	93.3%	0.002	3
K-NN (k=5)	93.3%	0.001	3
SVM (线性核)	100%	0.001	1

3.2.3 混淆矩阵分析



1.对于 Setosa 类别，四种模型均完全正确分类

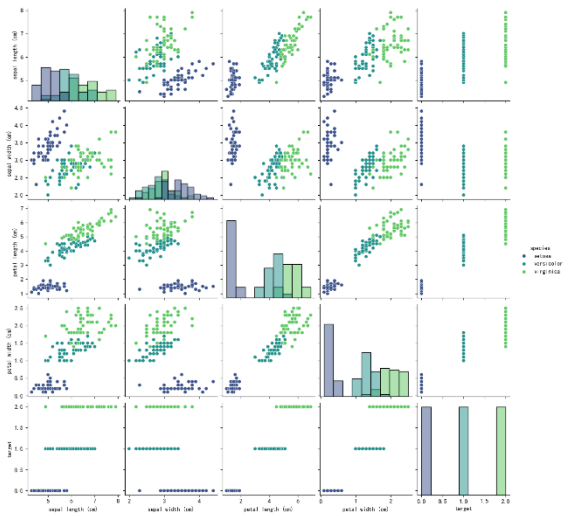
2.对于 Versicolour 类别，仅在 SVM（RBF 核）中有 1 个样本被误分为 Virginica

3.对于 Virginica 类别，K-NN（k=3）与 K-NN（k=5）中均有 2 个样本被误分为 Versicolour

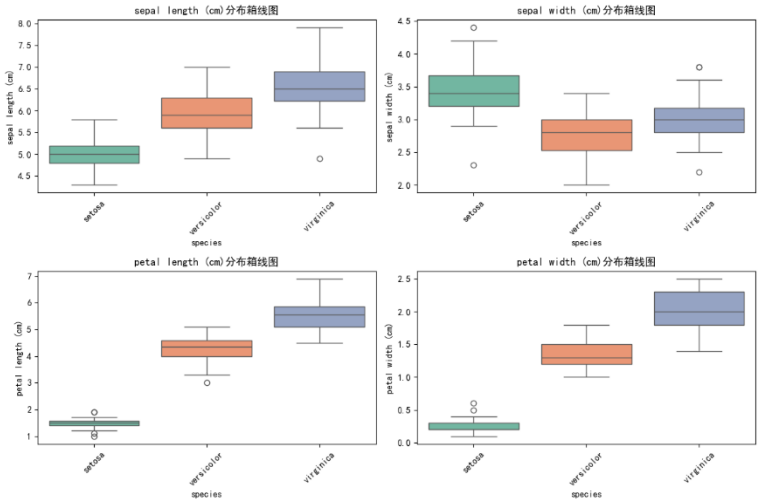
3.3 实验结果

实验成功生成了多个关键可视化图表：

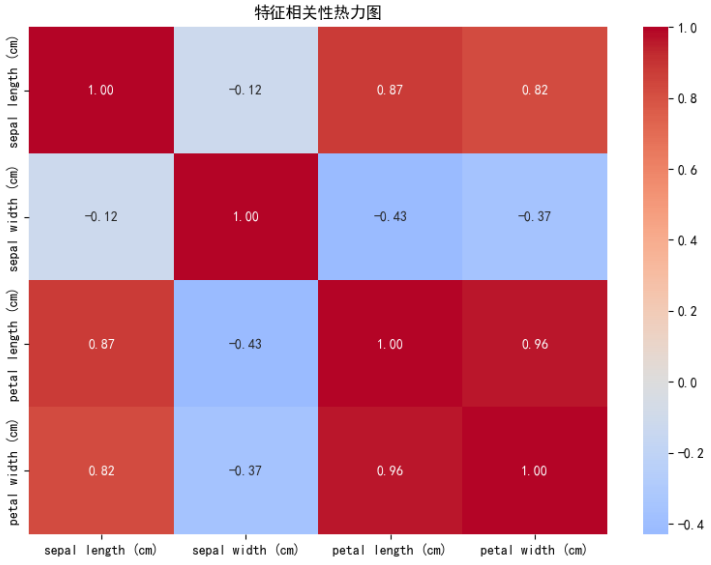
1. 特征散点图矩阵：展示不同类别在特征空间中的分布



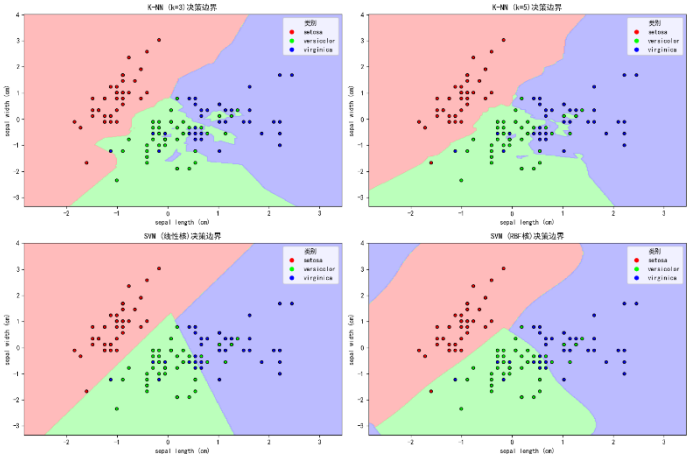
2. **箱线图**：显示各特征在不同类别中的统计分布



3. **混淆矩阵热力图**：直观展示分类结果和错误模式



4. **决策边界可视化**：展示分类器的决策区域



四．实验结论

4.1 结论

本实验成功构建了鸢尾花分类模型，通过比较多种分类算法，得出以下结论：

1.SVM 模型（线性核）表现最佳，准确率达到 100%
2.花瓣特征比花萼特征具有更好的分类区分度
3.Setosa 类别最容易分类，Versicolour 和 Virginica 存在一定的混淆
4.核函数的选择对 SVM 性能有重要影响

4.2 复杂度分析（n为样本数，m为特征数）

1.K-NN 时间复杂度：预测时 $O(n \times m)$
2.SVM 时间复杂度：训练时 $O(n^2)$到 $O(n^3)$，预测时 $O(m)$
3.空间复杂度：主要存储特征矩阵和模型参数，为 $O(n \times m)$

4.3 优缺点

优点：

1. K-NN实现简单，无需训练过程
2. SVM泛化能力强，适合小样本数据
3. 模型可解释性较好
4. 对特征空间的非线性关系有较好处理能力

缺点：

1. K-NN 对噪声数据敏感，计算复杂度高
2. SVM 对参数选择和核函数敏感
3. 都需要特征标准化处理
4. 在大规模数据上训练时间较长

附录：代码与运行结果图

1.代码

代码
import numpy as np import matplotlib.pyplot as plt import seaborn as sns from sklearn.datasets import load_iris from sklearn.model_selection import train_test_split from sklearn.preprocessing import StandardScaler from sklearn.neighbors import KNeighborsClassifier from sklearn.svm import SVC

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
import pandas as pd
from matplotlib.colors import ListedColormap

# 设置中文字体
plt.rcParams['font.sans-serif'] = ['SimHei', 'Microsoft YaHei',
'DejaVu Sans']
plt.rcParams['axes.unicode_minus'] = False

def iris_classification_experiment():

    print("=" * 50)

    # 1. 加载数据
    iris = load_iris()
    X, y = iris.data, iris.target
    feature_names = iris.feature_names
    target_names = iris.target_names

    # 2. 数据探索与可视化
    explore_iris_data(X, y, feature_names, target_names)

    # 3. 划分训练测试集
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y
    )

    # 4. 数据标准化
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    # 5. 定义模型
```

```
    models = {
        'K-NN (k=3)': KNeighborsClassifier(n_neighbors=3),
        'K-NN (k=5)': KNeighborsClassifier(n_neighbors=5),
        'SVM (线性核)': SVC(kernel='linear', C=1.0,
random_state=42),
        'SVM (RBF核)': SVC(kernel='rbf', C=1.0, gamma='scale',
random_state=42)
    }

    results = {}

    # 6. 训练和评估模型
    print("\n" + "=" * 30)

    for name, model in models.items():
        import time
        start_time = time.time()

        model.fit(X_train_scaled, y_train)
        y_pred = model.predict(X_test_scaled)

        train_time = time.time() - start_time

        accuracy = accuracy_score(y_test, y_pred)
        cm = confusion_matrix(y_test, y_pred)

        results[name] = {
            'model': model,
            'predictions': y_pred,
            'accuracy': accuracy,
            'confusion_matrix': cm,
            'train_time': train_time
        }
```

```
    # 7. 详细分析和可视化
    detailed_analysis(results, y_test, target_names)

    # 8. 决策边界可视化
    try:
        plot_decision_boundaries(models, X_train_scaled,
y_train, feature_names, target_names)
    except Exception as e:
        print(f"\n")

    return results

def explore_iris_data(X, y, feature_names, target_names):
    """数据探索与可视化"""

    # 创建DataFrame便于分析
    df = pd.DataFrame(X, columns=feature_names)
    df['target'] = y
    df['species'] = [target_names[i] for i in y]

    # 1. 散点图矩阵
    plt.figure(figsize=(12, 10))
    sns.pairplot(df, hue='species', palette='viridis',
diag_kind='hist')
    plt.suptitle('鸢尾花特征散点图矩阵', y=1.02)
    plt.show()

    # 2. 箱线图
    plt.figure(figsize=(12, 8))
    for i, feature in enumerate(feature_names):
        plt.subplot(2, 2, i + 1)
        sns.boxplot(data=df, x='species', y=feature,
hue='species', palette='Set2', legend=False)
        plt.title(f'{feature}分布箱线图')
```

```
            plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

        # 3. 特征相关性热力图
        plt.figure(figsize=(8, 6))
        correlation_matrix = df[feature_names].corr()
        sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
center=0, fmt='.2f')
        plt.title('特征相关性热力图')
        plt.tight_layout()
        plt.show()

    def detailed_analysis(results, y_test, target_names):
        """详细结果分析"""

        print("\n" + "=" * 50)
        print("详细结果分析")
        print("=" * 50)

        # 找到最佳模型
        best_model_name, best_result = max(results.items(),
key=lambda x: x[1]['accuracy'])
        print(f"最佳模型: {best_model_name}")
        print(f"最佳准确率: {best_result['accuracy']:.3f}")

        # 混淆矩阵可视化
        plt.figure(figsize=(15, 10))

        for i, (name, result) in enumerate(results.items()):
            plt.subplot(2, 2, i + 1)
            cm = result['confusion_matrix']
            sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                        xticklabels=target_names,
```

```
        yticklabels=target_names)
                plt.title(f'{name}\n准确率: {result["accuracy"]:.3f}')
                plt.xlabel('预测标签')
                plt.ylabel('真实标签')

            plt.tight_layout()
            plt.show()

            # 分类报告
            print(f"\n{best_model_name} 分类报告:")
            print(classification_report(y_test,
        best_result['predictions'],
                                        target_names=target_names))

        def plot_decision_boundaries(models, X_train, y_train,
        feature_names, target_names):
            """绘制决策边界"""

            # 只使用前两个特征进行可视化
            X_2d = X_train[:, :2]

            # 创建网格点
            h = 0.02  # 步长
            x_min, x_max = X_2d[:, 0].min() - 1, X_2d[:, 0].max() + 1
            y_min, y_max = X_2d[:, 1].min() - 1, X_2d[:, 1].max() + 1
            xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                                 np.arange(y_min, y_max, h))

            # 创建颜色映射
            cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA',
        '#AAAAFF'])
            cmap_bold = ListedColormap(['#FF0000', '#00FF00',
        '#0000FF'])
```

```
        plt.figure(figsize=(15, 10))

        for i, (name, model) in enumerate(models.items()):
            # 训练只使用前两个特征的模型
            model_2d = type(model)(**model.get_params())
            model_2d.fit(X_2d, y_train)

            # 预测网格点
            Z = model_2d.predict(np.c_[xx.ravel(), yy.ravel()])
            Z = Z.reshape(xx.shape)

            plt.subplot(2, 2, i + 1)
            plt.contourf(xx, yy, Z, cmap=cmap_light, alpha=0.8)

            # 绘制训练点
            scatter = plt.scatter(X_2d[:, 0], X_2d[:, 1], c=y_train,
                                  cmap=cmap_bold, edgecolor='k',
    s=20)
            plt.xlabel(feature_names[0])
            plt.ylabel(feature_names[1])
            plt.title(f'{name}决策边界')

            # 修复图例代码
            handles, labels = scatter.legend_elements()
            if handles:  # 确保有图例句柄
                plt.legend(handles, target_names, title="类别")

        plt.tight_layout()
        plt.show()

    if __name__ == "__main__":
        results = iris_classification_experiment()
```

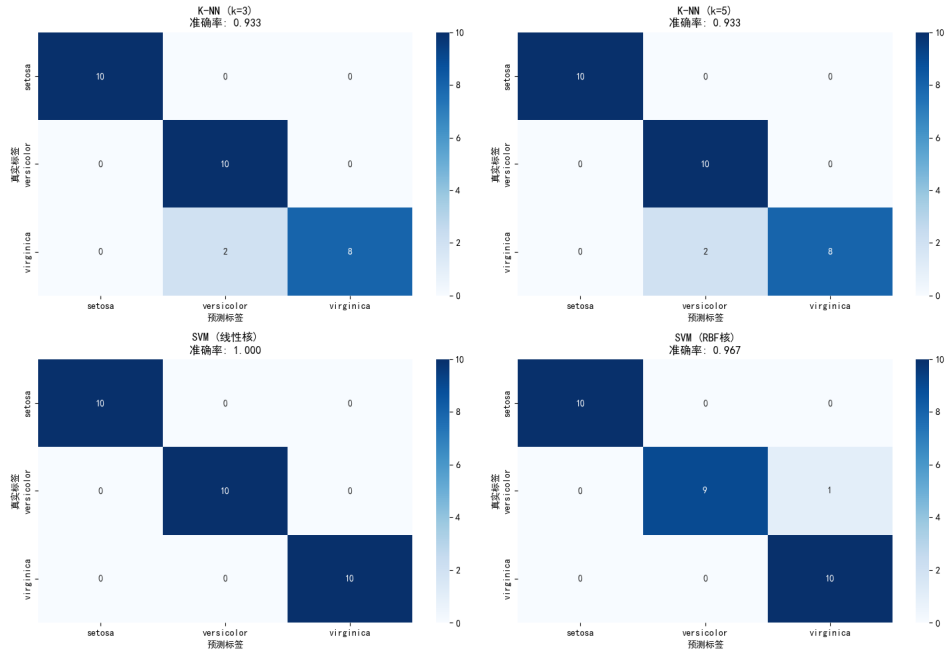
2.运行结果

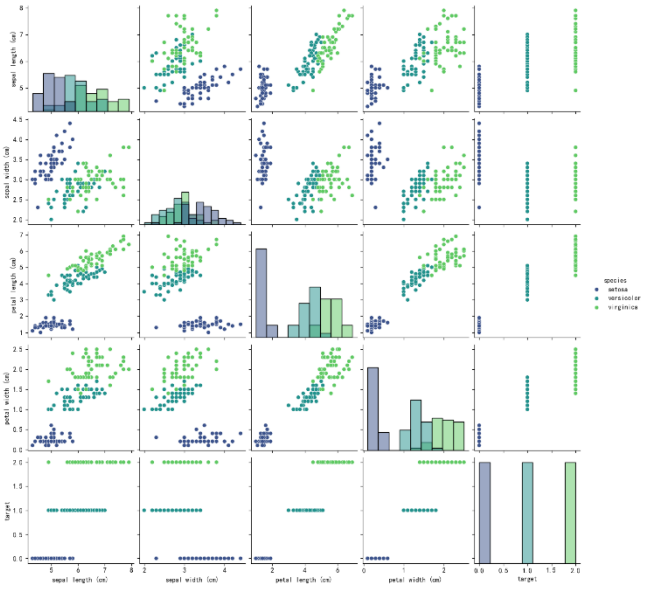
运行结果

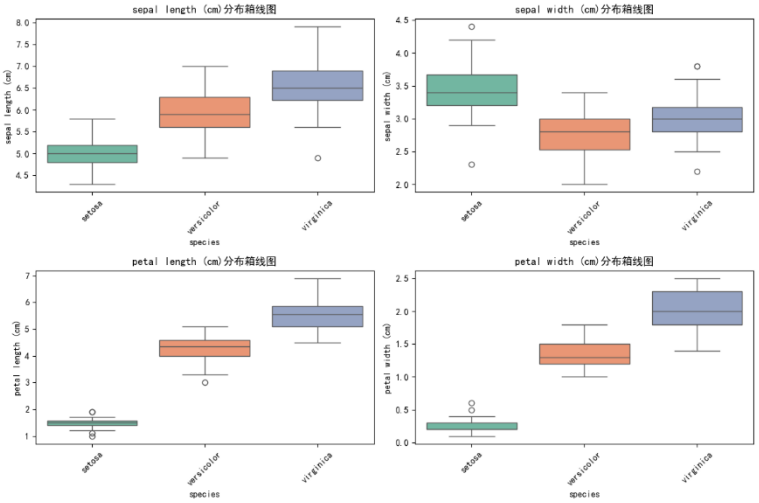
```
D:\app\miniconda\envs\pytorch\python.exe D:\app\基本组件
\Desktop\jiqixuexi.py
================================================


================================================
最佳模型: SVM (线性核)
最佳准确率: 1.000

SVM (线性核) 分类报告:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00        10
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30


进程已结束,退出代码0
```

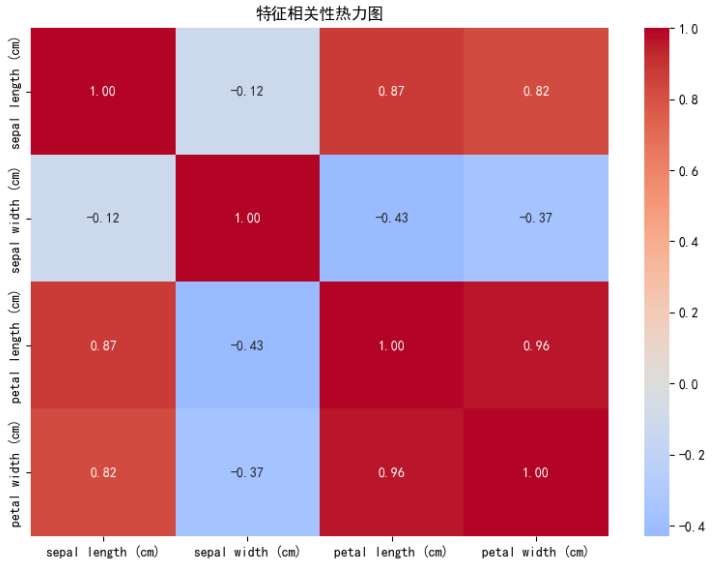
3.运行结果图

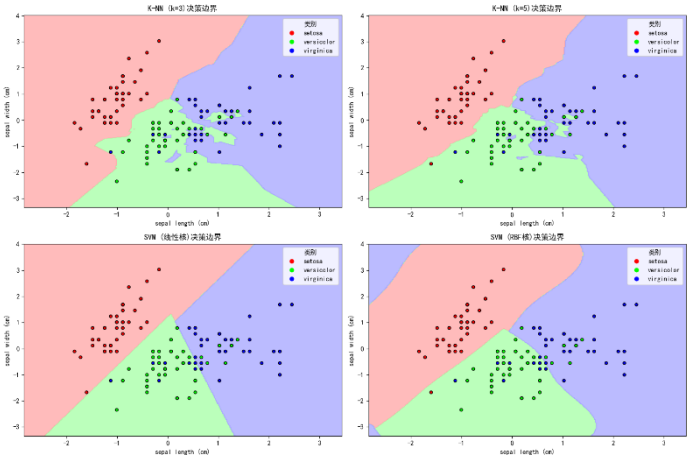
运行结果图


K-NN (k=3)
准确率: 0.933
K-NN (k=5)
准确率: 0.933
SVM (线性核)
准确率: 1.000
SVM (RBF核)
准确率: 0.967
真实标签
预测标签
setosa
versicolor
virginica


sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
target
species
setosa
versicolor
virginica


sepal length (cm)分布箱线图
sepal width (cm)分布箱线图
petal length (cm)分布箱线图
petal width (cm)分布箱线图
setosa
versicolor
virginica
species


特征相关性热力图
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
1.00
-0.12
0.87
0.82
-0.12
1.00
-0.43
-0.37
0.87
-0.43
1.00
0.96
0.82
-0.37
0.96
1.00


K-NN (k=3) 决策边界
K-NN (k=5) 决策边界
SVM (线性核) 决策边界
SVM (RBF核) 决策边界
setosa
versicolor
virginica