**《数据统计与分析基础实验》**

实验四报告

|  |  |
| --- | --- |
| 班级： | 计224 |
| 姓名： | 王小康 |
| 学号： | 225432 |

**实验四 机器学习算法建模与求解**

**一、实验目的**

掌握使用实用软件通过各类基础的机器学习算法解决实际数据统计分析任务的能力，熟悉线性回归、SVM、kmeans、PCA 等算法的调用。

**二、实验内容**

1、对于下表中的数据，对1990年-2005年内的数据建立人口自然增长率对于国民总收入、CPI增长率和人均GDP的三元线性回归模型。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 年份 | 人口自然增长率y/% | 国民总收入x1/亿元 | CPI增长率x2/% | 人均GDPx3/% |
| 1988 | 15.73 | 15037 | 18.8 | 1366 |
| 1989 | 15.04 | 17001 | 18 | 1519 |
| 1990 | 14.39 | 18718 | 3.1 | 1644 |
| 1991 | 12.98 | 21826 | 3.4 | 1893 |
| 1992 | 11.6 | 26937 | 6.4 | 2311 |
| 1993 | 11.45 | 35260 | 14.7 | 2998 |
| 1994 | 11.21 | 48108 | 24.1 | 4044 |
| 1995 | 10.55 | 59811 | 17.1 | 5046 |
| 1996 | 10.42 | 70142 | 8.3 | 5846 |
| 1997 | 10.06 | 78061 | 2.8 | 6420 |
| 1998 | 9.14 | 83024 | -0.8 | 6796 |
| 1999 | 8.18 | 88479 | -1.4 | 7159 |
| 2000 | 7.58 | 98000 | 0.4 | 7858 |
| 2001 | 6.95 | 108068 | 0.7 | 8622 |
| 2002 | 6.45 | 119096 | -0.8 | 9398 |
| 2003 | 6.01 | 135174 | 1.2 | 10542 |
| 2004 | 5.87 | 159587 | 3.9 | 12336 |
| 2005 | 5.89 | 184089 | 1.8 | 14040 |
| 2006 | 5.38 | 213132 | 1.5 | 16024 |

2、下载UCI中wine数据集：http://archive.ics.uci.edu/ml/datasets/Wine。所下载数据可以用txt打开，其中每一行数据为一种Wine的记录，每条记录包含14个维度，其中第一维为该Wine类别，后面13维为具体的Wine属性。请基于所有的178个Wine样本对Wine的13个维度进行PCA降维分析，将贡献率之和大于90%的成分提取，并将原13维属性数据映射为新数据。

3、（1）随机生成均值、方差各不相同，且相互之间有少量交叉的3个类，每类30个样本，用不同的颜色进行展示。

（2）通过kmeans聚类分析，将所有的数据分成3类、4类、5类，每一类用不同颜色展示。

共形成4张图。

4、随机生成完全不交叉的2个类，每个类包含30个样本，用SVM进行分类和返回所有支撑向量，并以合适方式进行Figure展示。

**三、实验源程序及运行结果（可附截图）**

1.程序如下：

import pandas as pd

from sklearn.linear\_model import LinearRegression

# 数据（1990-2005）

data = {

    'Year': [1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005],

    'Population Growth Rate': [14.39, 12.98, 11.6, 11.45, 11.21, 10.55, 10.42, 10.06, 9.14, 8.18, 7.58, 6.95, 6.45, 6.01, 5.87, 5.89],

    'GDP': [18718, 21826, 26937, 35260, 48108, 59811, 70142, 78061, 83024, 88479, 98000, 108068, 119096, 135174, 159587, 184089],

    'CPI Growth Rate': [3.1, 3.4, 6.4, 14.7, 24.1, 17.1, 8.3, 2.8, -0.8, -1.4, 0.4, 0.7, -0.8, 1.2, 3.9, 1.8],

    'Per Capita GDP': [1644, 1519, 1644, 1893, 2311, 2998, 4044, 5046, 5846, 6420, 6796, 7159, 7858, 8622, 9398, 10542]

}

# 创建 DataFrame

df = pd.DataFrame(data)

# 提取自变量 X 和因变量 y

X = df[['GDP', 'CPI Growth Rate', 'Per Capita GDP']]

y = df['Population Growth Rate']

# 线性回归

model = LinearRegression()

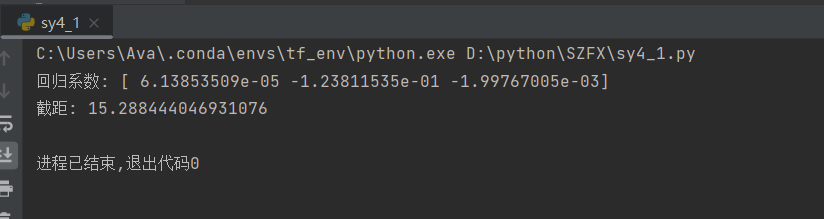
model.fit(X, y)

# 显示回归系数和截距

print("回归系数:", model.coef\_)

print("截距:", model.intercept\_)

结果如下：



2.程序如下：

import pandas as pd

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

import numpy as np

# 加载数据，指定列名

columns = ['Class', 'Alcohol', 'Malic Acid', 'Ash', 'Alcalinity of Ash', 'Magnesium',

           'Total Phenols', 'Flavanoids', 'Nonflavanoid Phenols', 'Proanthocyanins',

           'Color Intensity', 'Hue', 'OD280/OD315 of Diluted Wines', 'Proline']

# 读取 Wine 数据集

data = pd.read\_csv(r'D:\python\SZFX\wine.data', header=None, names=columns)

# 提取特征数据（排除类别标签列）

X = data.iloc[:, 1:].values

# 数据标准化

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# 进行 PCA 降维，保留所有主成分

pca = PCA(n\_components=13)  # 保留所有成分

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

# 查看各主成分的贡献率

explained\_variance = pca.explained\_variance\_ratio\_

print(f"各主成分的贡献率: {explained\_variance}")

# 查看累计贡献率

cumulative\_variance = np.cumsum(explained\_variance)

print(f"累计贡献率: {cumulative\_variance}")

# 找到贡献率之和大于 90%的成分

n\_components = np.argmax(cumulative\_variance >= 0.90) + 1

print(f"保留的主成分数: {n\_components}")

# 提取贡献率和大于 90% 的主成分

pca = PCA(n\_components=n\_components)

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

# 输出降维后的数据

print(f"降维后的数据:\n{X\_pca\_reduced}")

结果如下：

C:\Users\Ava\.conda\envs\tf\_env\python.exe D:\python\SZFX\sy4\_2.py

各主成分的贡献率: [0.36198848 0.1920749 0.11123631 0.0706903 0.06563294 0.04935823

0.04238679 0.02680749 0.02222153 0.01930019 0.01736836 0.01298233

0.00795215]

累计贡献率: [0.36198848 0.55406338 0.66529969 0.73598999 0.80162293 0.85098116

0.89336795 0.92017544 0.94239698 0.96169717 0.97906553 0.99204785

1. ]

保留的主成分数: 8

降维后的数据:

[[ 3.31675081 1.44346263 -0.16573904 ... 0.22388013 0.59642655

-0.06513909]

[ 2.20946492 -0.33339289 -2.02645737 ... 0.92712024 0.05377561

-1.02441595]

[ 2.51674015 1.0311513 0.98281867 ... -0.54927605 0.42420545

0.34421613]

...

[-2.67783946 2.76089913 -0.94094188 ... -0.27306792 0.67923541

-0.0470238 ]

[-2.38701709 2.29734668 -0.5506962 ... -1.17878298 0.63397527

-0.39082877]

[-3.20875816 2.76891957 1.01391366 ... -0.29609218 0.00574126

0.29291373]]

进程已结束,退出代码0

3（1）.程序如下：

import numpy as np

import matplotlib.pyplot as plt

# 设置随机种子，保证结果可复现

np.random.seed(42)

# 生成每个类的数据（30个样本），每个类的均值和方差不同，且有所交叉

# 类别1: 均值 (3, 3)，方差较小

class1 = np.random.normal(loc=[3, 3], scale=[1, 1], size=(30, 2))

# 类别2: 均值 (5, 5)，方差较大，与类别1略有交叉

class2 = np.random.normal(loc=[5, 5], scale=[2, 2], size=(30, 2))

# 类别3: 均值 (7, 7)，方差较大，与类别2略有交叉

class3 = np.random.normal(loc=[7, 7], scale=[1.5, 1.5], size=(30, 2))

# 合并所有类别的数据

X = np.vstack([class1, class2, class3])

y = np.array([0] \* 30 + [1] \* 30 + [2] \* 30)  # 类别标签

# 绘制数据点

plt.figure(figsize=(8, 6))

# 绘制每个类别的数据点

plt.scatter(class1[:, 0], class1[:, 1], color='red', label='Class 1', alpha=0.6)

plt.scatter(class2[:, 0], class2[:, 1], color='blue', label='Class 2', alpha=0.6)

plt.scatter(class3[:, 0], class3[:, 1], color='green', label='Class 3', alpha=0.6)

# 设置标题和标签

plt.title('Generated Data with 3 Classes')

plt.xlabel('Feature 1')

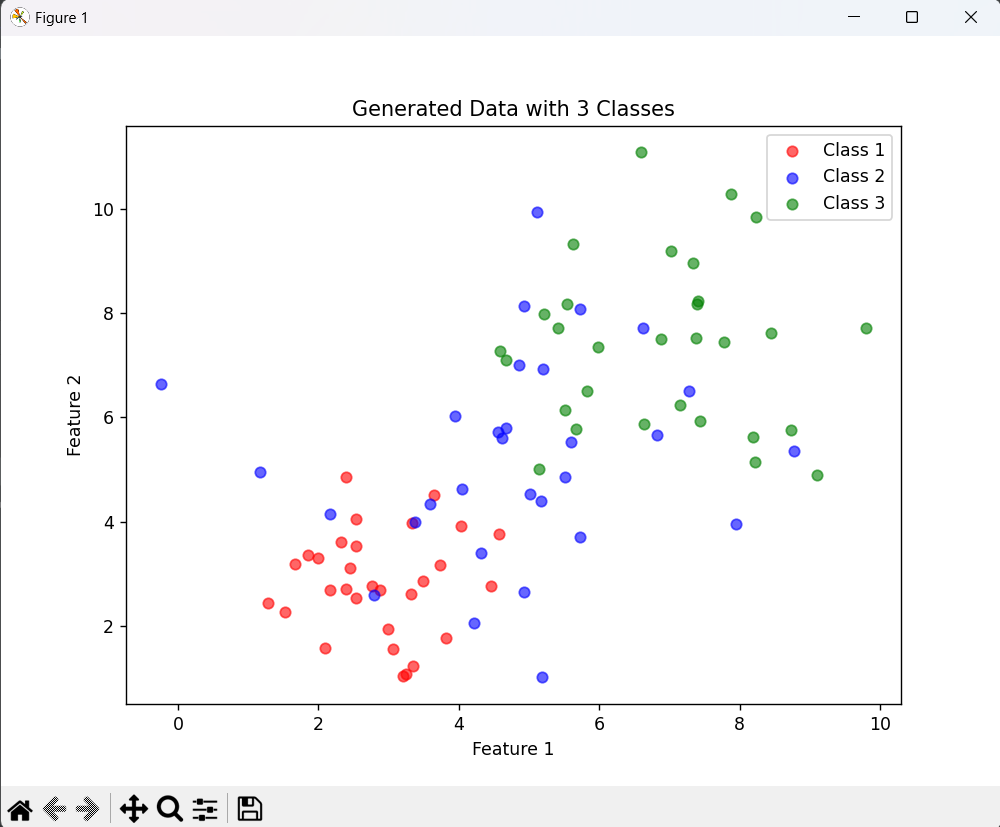
plt.ylabel('Feature 2')

plt.legend()

# 显示图像

plt.show()

结果如下：



3（2）.程序如下：

from sklearn.cluster import KMeans

import numpy as np

import matplotlib.pyplot as plt

# 设置随机种子，保证结果可复现

np.random.seed(42)

# 生成每个类的数据（30个样本），每个类的均值和方差不同，且有所交叉

class1 = np.random.normal(loc=[3, 3], scale=[1, 1], size=(30, 2))

class2 = np.random.normal(loc=[5, 5], scale=[2, 2], size=(30, 2))

class3 = np.random.normal(loc=[7, 7], scale=[1.5, 1.5], size=(30, 2))

# 合并所有类别的数据

X = np.vstack([class1, class2, class3])

# 定义绘图函数

def plot\_kmeans(X, k, ax):

    kmeans = KMeans(n\_clusters=k, random\_state=42)

    y\_kmeans = kmeans.fit\_predict(X)

    # 绘制每个聚类的数据点

    ax.scatter(X[:, 0], X[:, 1], c=y\_kmeans, cmap='viridis', alpha=0.6)

    # 绘制聚类中心

    ax.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=200, c='red', marker='X',

               label='Centroids')

    ax.set\_title(f'KMeans Clustering with {k} clusters')

    ax.set\_xlabel('Feature 1')

    ax.set\_ylabel('Feature 2')

    ax.legend()

# 创建子图

fig, axs = plt.subplots(1, 3, figsize=(18, 6))

# 对数据进行3类、4类和5类的KMeans聚类并绘制结果

plot\_kmeans(X, 3, axs[0])

plot\_kmeans(X, 4, axs[1])

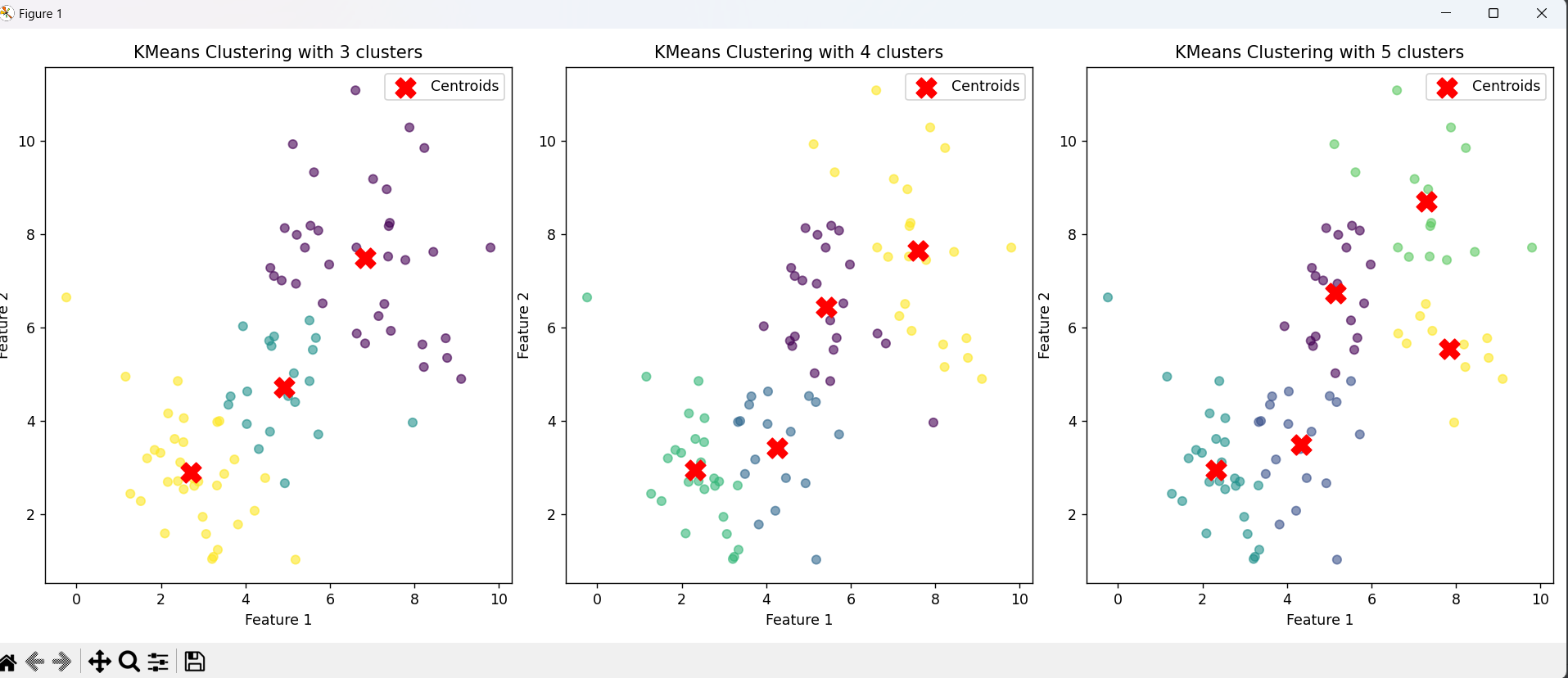
plot\_kmeans(X, 5, axs[2])

# 显示图像

plt.tight\_layout()

plt.show()

结果如下：



4.程序如下：

import numpy as np

import matplotlib.pyplot as plt

from sklearn import svm

# 设置随机种子，保证结果可复现

np.random.seed(42)

# 生成两个类别的数据，每个类包含30个样本，且完全不交叉

class1 = np.random.normal(loc=[2, 2], scale=[0.5, 0.5], size=(30, 2))  # 类别1，均值(2, 2)

class2 = np.random.normal(loc=[6, 6], scale=[0.5, 0.5], size=(30, 2))  # 类别2，均值(6, 6)

# 合并所有类别的数据

X = np.vstack([class1, class2])

y = np.array([0] \* 30 + [1] \* 30)  # 类别标签

# 创建并训练SVM模型

clf = svm.SVC(kernel='linear')

clf.fit(X, y)

# 获取所有支撑向量

support\_vectors = clf.support\_vectors\_

# 绘制数据点

plt.figure(figsize=(8, 6))

# 绘制类别1和类别2的数据点

plt.scatter(class1[:, 0], class1[:, 1], color='red', label='Class 1', alpha=0.6)

plt.scatter(class2[:, 0], class2[:, 1], color='blue', label='Class 2', alpha=0.6)

# 绘制支撑向量

plt.scatter(support\_vectors[:, 0], support\_vectors[:, 1], color='green', marker='X', s=150, label='Support Vectors')

# 绘制决策边界

xx, yy = np.meshgrid(np.linspace(-1, 8, 100), np.linspace(-1, 8, 100))

Z = clf.decision\_function(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

# 绘制决策边界和边界间隔

plt.contour(xx, yy, Z, levels=[0], linewidths=2, colors='black')

plt.contour(xx, yy, Z, levels=[-1, 1], linewidths=1, colors='black', linestyles='dashed')

# 设置标题和标签

plt.title('SVM Classification with Support Vectors')

plt.xlabel('Feature 1')

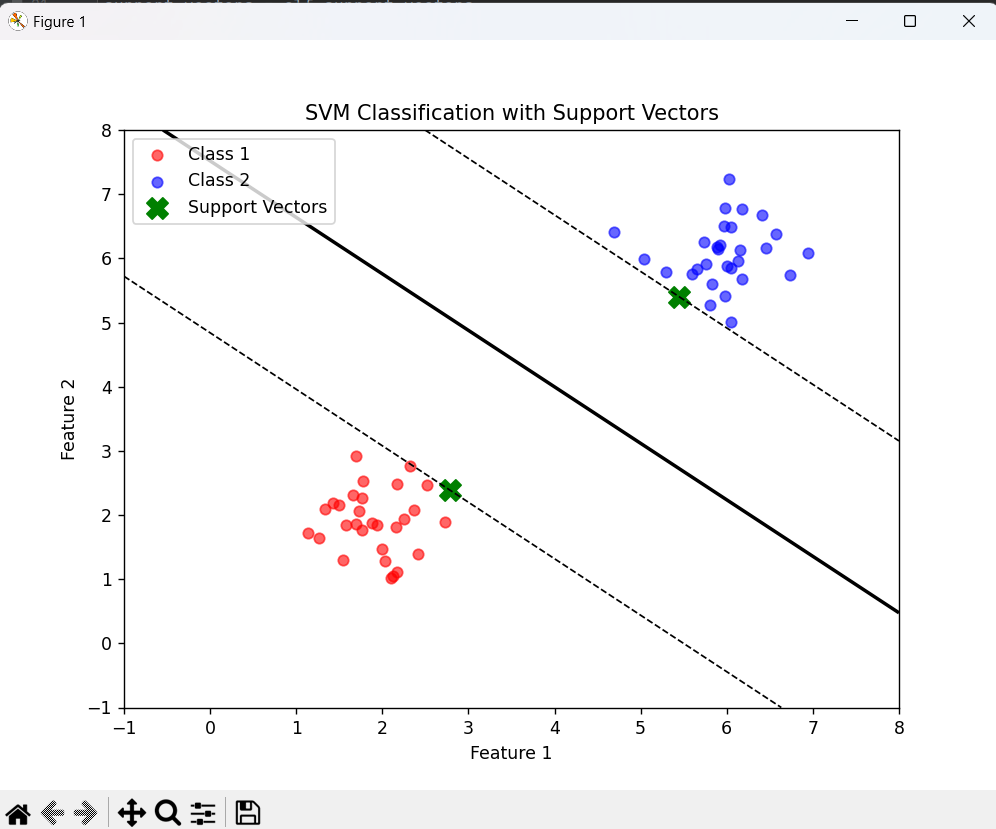
plt.ylabel('Feature 2')

plt.legend()

# 显示图像

plt.show()

结果如下：



**四、实验感想**

#### 1. ****人口自然增长率与国民总收入、CPI增长率和人均GDP的三元线性回归模型****

通过这个实验，我深入理解了回归模型在现实数据分析中的应用。在建立三元线性回归模型时，我们考虑了三个自变量（国民总收入、CPI增长率和人均GDP）与因变量（人口自然增长率）之间的关系。这使我意识到，统计模型不仅能够帮助我们分析变量之间的相关性，还可以帮助预测未知数据。在处理实际问题时，回归模型的建立不仅依赖于理论假设，还需要数据的质量和量的支持。此外，回归分析中的残差分析和多重共线性等问题也是我需要继续关注的内容。

#### 2. ****UCI Wine 数据集的 PCA 降维分析****

PCA（主成分分析）降维的实验给我提供了一个很好的数据降维的实践机会。在这个实验中，我首先通过对原始13个特征进行降维，提取出贡献率累计超过90%的主成分。PCA是一种强有力的数据分析工具，能够有效地降低数据的维度，同时保持大部分信息。在分析过程中，我意识到高维数据可能带来计算上的复杂性，而降维不仅能简化计算，还可以在可视化和进一步的机器学习任务中发挥重要作用。通过降维分析，我可以更清晰地理解数据背后的结构，发现潜在的规律。

#### 3. ****K-means 聚类分析****

K-means聚类实验让我进一步加深了对无监督学习方法的理解。在处理生成的样本数据时，我通过调整聚类的数量（3类、4类、5类）来探索数据的不同结构。这个过程不仅帮助我掌握了如何使用K-means算法对数据进行分类，还让我感受到聚类数目的选择对结果的重要性。通过可视化分析，我能够看到不同聚类数下的分类效果，并意识到在实际应用中，如何选择合适的聚类数是一个非常具有挑战性的问题。这个实验让我更加熟悉了K-means算法的应用，并且对无监督学习的优缺点有了更深刻的理解。

#### 4. ****SVM 分类与支持向量展示****

最后，SVM实验让我对有监督学习中的分类方法有了更深入的理解。支持向量机不仅可以处理线性可分的数据，而且通过选择合适的核函数，能够解决非线性分类问题。在实验中，生成了两个完全不交叉的类别数据，通过SVM进行分类，展示了如何用决策边界区分不同类别的样本，并找出了所有的支撑向量。这些支撑向量是SVM模型的关键，它们决定了分类超平面的最优位置。这个实验不仅让我理解了SVM的基本原理，还让我对分类边界的可视化及模型的优化有了更直观的认识。

### 总结

这些实验帮助我更加深入地理解了机器学习中的几种基本方法：回归分析、PCA降维、K-means聚类和SVM分类。每个实验都有其独特的应用场景和理论背景，而通过实际操作，我也体会到了不同算法在处理不同问题时的优劣和适用性。未来，我会更加注重这些方法的实际应用，并尝试将它们结合起来解决更复杂的实际问题。