**南京信息工程大学 人工智能**导论 **实验(实习)报告**

实验(实习)名称 人工智能导论 实验(实习)日期 2024.11.07 得分 指导教师 应龙

学院 计算机与网络空间安全学院专业 计算机科学与技术 年级 2022 班次 4 姓名张瑞晨 学号 202283290159

1. 实验目的

1. 掌握 kmeans聚类算法的主要思想和优化目标函数。

2. 掌握 kmeans聚类的算法步骤。

3.了解聚类的评价指标：基于有标签测试数据的评价指标和无标签测试数据的评价指标。

4.学习传统机器学习框架基于特征的人脸表示方法。

5. 学习scikit 库中PCA相关模块的使用和基于这些模块的特征脸表示。

6. 学习scikit 库中SVM相关模块的使用。

7. 了解一些多类别分类评价指标。

二、实验内容与步骤

1.K-means聚类

K-means聚类是一种非常流行且广泛使用的无监督学习算法，主要用于将数据集中的样本划分成K个聚类，其中每个样本都属于最近的均值（即聚类中心）所代表的聚类。K-means算法的目标是最小化每个点到其聚类中心的距离的总和，这个距离通常是欧几里得距离。

K-means算法的基本步骤：

初始化：

随机选择K个数据点作为初始的聚类中心（质心）。

分配：

将每个数据点分配给最近的聚类中心，形成K个聚类。

更新：

重新计算每个聚类的中心，即每个聚类中所有点的均值。

迭代：

重复步骤2和3，直到满足某个终止条件，如聚类中心的变化小于一个预设的阈值，或者达到预设的迭代次数。

终止：

当满足终止条件时，算法结束，每个数据点都有了所属的聚类。

K-means算法的关键点：

K的选择：K的值需要预先指定，且对结果有很大影响。太小可能导致多个数据点被分到同一个聚类中，太大则可能导致每个数据点都单独成为一个聚类。常用的方法包括肘部法则（Elbow Method）、轮廓分析（Silhouette Analysis）等来估计最佳的K值。

聚类中心的初始化：聚类中心的初始选择对最终结果有影响，因为K-means算法可能会收敛到局部最优解。为了减少这种影响，可以通过多次运行算法并选择最佳结果来提高稳定性。

距离度量：K-means通常使用欧几里得距离，但也可以使用其他距离度量，如曼哈顿距离等。

收敛性：K-means算法是迭代的，并且通常会收敛，但不保证找到全局最优解。

对噪声和异常值的敏感性：K-means对噪声和异常值比较敏感，因为它们可能会对聚类中心产生较大影响。

缩放敏感性：由于K-means依赖于距离计算，因此对数据的缩放敏感。在应用K-means之前，通常需要对数据进行归一化处理。

K-means的应用场景：

K-means聚类因其简单和高效而被广泛应用于市场细分、社交网络分析、图像分割、文档分类等领域。尽管它有一些局限性，但由于其计算效率和实现简单，K-means仍然是许多领域中首选的聚类方法之一。

**代码步骤：**

1. 导入必要的库：matplotlib.cm、matplotlib.pyplot、numpy、sklearn.cluster中的KMeans、sklearn.datasets中的make\_blobs、sklearn.metrics中的silhouette\_samples和silhouette\_score。

2.使用make\_blobs函数生成样本数据，这里生成了500个样本，2个特征，4个中心点，并且设置了一个随机状态以保证结果的可重复性。

3.定义了一个聚类数的范围range\_n\_clusters，包含了从2到6的聚类数。

4.对于每个聚类数，创建一个子图，其中包含两个轴：一个用于绘制轮廓图，另一个用于绘制实际的聚类结果。

5.对于每个聚类数，初始化一个KMeans聚类器，并使用fit\_predict方法对数据进行聚类。

6.计算并打印所有样本的平均轮廓系数。

7.计算每个样本的轮廓分数，并根据聚类标签对它们进行排序。

8.在轮廓图中，为每个聚类绘制填充区域，并在中间位置标记聚类编号。

9.在轮廓图中添加平均轮廓系数的垂直线。

10.在第二个子图中，使用散点图显示聚类结果，并在聚类中心绘制白色的圆圈。

11.设置子图的标题和轴标签。

12.使用plt.show()显示所有的子图。

**用了几个关键的函数和类，每个都有其特定的功能和输入输出。以下是它们的总结：**

make\_blobs：

功能：生成指定数量的样本和特征的随机数据集，这些数据集围绕指定数量的中心点（聚类中心）分布。

输入：样本数量（n\_samples）、特征数量（n\_features）、中心点数量（centers）、中心点的方差（cluster\_std）、中心点的边界（center\_box）、是否打乱数据（shuffle）和随机状态（random\_state）。

输出：生成的数据集（X）和对应的标签（y）。

KMeans：

功能：KMeans聚类算法的实现，用于将数据集分成指定数量的聚类。

输入：聚类数（n\_clusters）、初始化次数（n\_init）和随机状态（random\_state）。

输出：聚类中心（cluster\_centers\_）和每个样本的聚类标签（cluster\_labels）。

silhouette\_score：

功能：计算所有样本的平均轮廓系数，用于评估聚类效果。

输入：数据集（X）和样本的聚类标签（cluster\_labels）。

输出：平均轮廓系数。

silhouette\_samples：

功能：计算每个样本的轮廓系数。

输入：数据集（X）和样本的聚类标签（cluster\_labels）。

输出：每个样本的轮廓系数数组。

plt.subplots：

功能：创建一个图形窗口，并在其中创建子图。

输入：子图的行数和列数。

输出：图形对象和子图的轴对象。

算法功能模块设计的重要思想：

数据可视化：通过轮廓图和散点图直观地展示聚类效果，使得非专业人士也能评估聚类算法的性能。

参数优化：使用轮廓分析来选择最佳的聚类数（n\_clusters），这是一种无监督学习方法中常见的参数优化手段。

模块化设计：代码将数据生成、聚类、评估和可视化分开处理，使得每个部分都可以独立使用和修改，提高了代码的可维护性和可扩展性。

重复性和可复现性：通过设置随机状态（random\_state），确保了结果的可重复性，这对于科学研究和实验是非常重要的。

性能评估：轮廓系数提供了一个量化的指标来评估聚类的质量，这比仅仅依赖于视觉评估要准确得多。

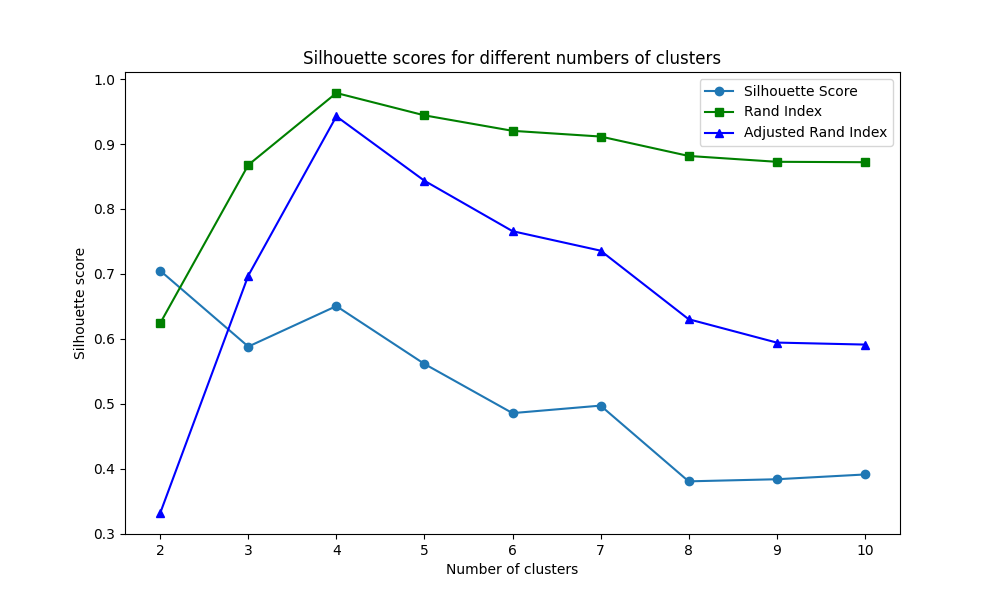
用户交互：虽然这个脚本是自动化运行的，但是它的设计允许用户通过修改输入参数来探索不同的聚类效果，增加了用户与程序的交互性。

**实验结果：**

**1.聚类簇数2-10的结果图像化**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. **2到10不同 k 值选取的 silhouette\_avg 曲线**



1. **对应的 Rand index 和 adjusted Rand index 的值。**

For n\_clusters = 2, silhouette\_avg: 0.705, Rand index: 0.624, Adjusted Rand index: 0.332

For n\_clusters = 3, silhouette\_avg: 0.588, Rand index: 0.868, Adjusted Rand index: 0.698

For n\_clusters = 4, silhouette\_avg: 0.651, Rand index: 0.979, Adjusted Rand index: 0.943

For n\_clusters = 5, silhouette\_avg: 0.561, Rand index: 0.944, Adjusted Rand index: 0.843

For n\_clusters = 6, silhouette\_avg: 0.486, Rand index: 0.920, Adjusted Rand index: 0.766

For n\_clusters = 7, silhouette\_avg: 0.497, Rand index: 0.912, Adjusted Rand index: 0.736

For n\_clusters = 8, silhouette\_avg: 0.381, Rand index: 0.882, Adjusted Rand index: 0.630

For n\_clusters = 9, silhouette\_avg: 0.384, Rand index: 0.873, Adjusted Rand index: 0.594

For n\_clusters = 10, silhouette\_avg: 0.391, Rand index: 0.872, Adjusted Rand index: 0.591

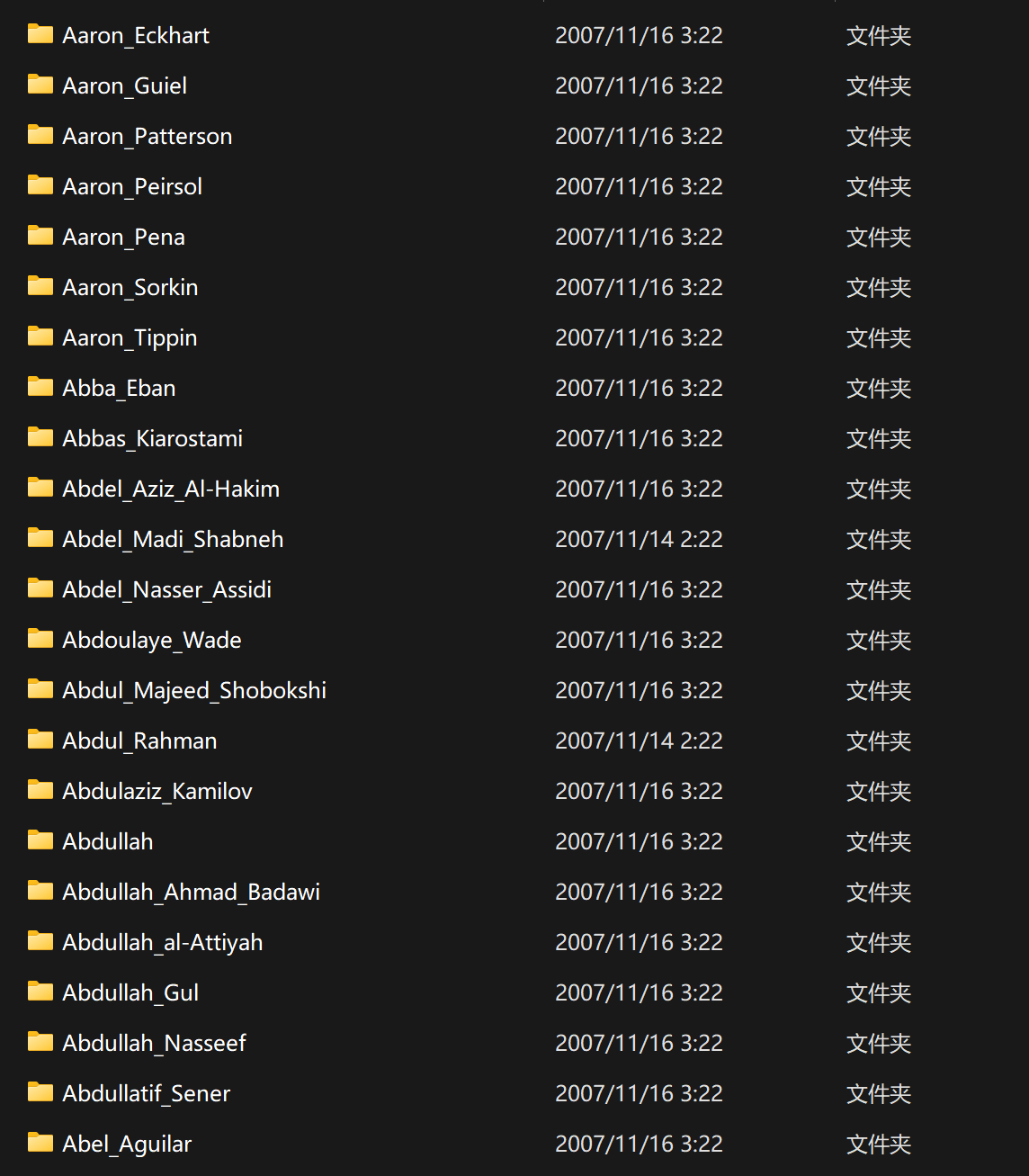
使用 **PCA（主成分分析）** 和 **支持向量机（SVM）** 来进行人脸识别的完整示例，使用的数据集为 **Labeled Faces in the Wild (LFW)**，该数据集包含了来自不同人的面部图像。

代码大致流程如下：

1. **导入必要的库**： 代码中使用了 sklearn 中的一些工具来进行数据加载、处理、建模和评估。同时，还使用了 matplotlib 用于图像的展示。
2. **加载数据**： 使用 fetch\_lfw\_people 函数加载经过预处理的人脸数据集，选取了至少有 70 张照片的人员数据。加载的数据包括：
   * 图像数据：每张图像的像素数据（lfw\_people.images）。
   * 标签数据：每个图像对应的标签（lfw\_people.target）。
3. **数据集分割与标准化**： 将数据分为训练集和测试集，训练集占 75%，测试集占 25%。使用 StandardScaler 对数据进行标准化，以确保每个特征的均值为 0，标准差为 1，这有助于模型训练的稳定性。
4. **PCA降维**： 使用 **PCA** 提取人脸数据的主成分（即特征），将数据从高维降至较低的维度。这里设置提取前 150 个主成分。PCA 过程中的 svd\_solver="randomized" 提供了一种更高效的随机化奇异值分解方法。
5. **训练支持向量机（SVM）**： 使用 **SVM** 进行分类，选择了 rbf 核函数，并通过 **随机搜索交叉验证（RandomizedSearchCV）** 来调整超参数 C 和 gamma。这有助于选择最佳的模型参数，以提高分类性能。
6. **模型评估**：
   * **定量评估**：通过 classification\_report 和 ConfusionMatrixDisplay 显示分类的详细报告和混淆矩阵，评估模型在测试集上的表现。
   * **定性评估**：将部分测试集图像与预测结果一起展示，以便更直观地了解模型的表现。
7. **可视化**：
   * 使用 plot\_gallery 函数显示测试集的预测结果，包括真实标签与预测标签的对比。
   * 显示最重要的特征“eigenfaces”（主成分），即提取出的面部特
   * 征。

**实验步骤：**

**1.首先解压数据集lfw-funneled.tgz，获得以文件夹名称形式记录的人脸记录**



**2.运行代码**"""

===================================================

Faces recognition example using eigenfaces and SVMs

===================================================

The dataset used in this example is a preprocessed excerpt of the

"Labeled Faces in the Wild", aka LFW\_:

  http://vis-www.cs.umass.edu/lfw/lfw-funneled.tgz (233MB)

.. \_LFW: http://vis-www.cs.umass.edu/lfw/

"""

# %%

from time import time

import matplotlib.pyplot as plt

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

from sklearn.model\_selection import RandomizedSearchCV

from sklearn.datasets import fetch\_lfw\_people

from sklearn.metrics import classification\_report

from sklearn.metrics import ConfusionMatrixDisplay

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.svm import SVC

from scipy.stats import loguniform  # 从 scipy.stats 导入 loguniform

# %%

# Download the data, if not already on disk and load it as numpy arrays

lfw\_people = fetch\_lfw\_people(min\_faces\_per\_person=70, resize=0.4)

# introspect the images arrays to find the shapes (for plotting)

n\_samples, h, w = lfw\_people.images.shape

# for machine learning we use the 2 data directly (as relative pixel

# positions info is ignored by this model)

X = lfw\_people.data

n\_features = X.shape[1]

# the label to predict is the id of the person

y = lfw\_people.target

target\_names = lfw\_people.target\_names

n\_classes = target\_names.shape[0]

print("Total dataset size:")

print("n\_samples: %d" % n\_samples)

print("n\_features: %d" % n\_features)

print("n\_classes: %d" % n\_classes)

# %%

# Split into a training set and a test and keep 25% of the data for testing.

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

    X, y, test\_size=0.25, random\_state=42

)

scaler = StandardScaler()

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

X\_test = scaler.transform(X\_test)

# %%

# Compute a PCA (eigenfaces) on the face dataset (treated as unlabeled

# dataset): unsupervised feature extraction / dimensionality reduction

n\_components = 150

print(

    "Extracting the top %d eigenfaces from %d faces" % (n\_components, X\_train.shape[0])

)

t0 = time()

pca = PCA(n\_components=n\_components, svd\_solver="randomized", whiten=True).fit(X\_train)

print("done in %0.3fs" % (time() - t0))

eigenfaces = pca.components\_.reshape((n\_components, h, w))

print("Projecting the input data on the eigenfaces orthonormal basis")

t0 = time()

X\_train\_pca = pca.transform(X\_train)

X\_test\_pca = pca.transform(X\_test)

print("done in %0.3fs" % (time() - t0))

# %%

# Train a SVM classification model

print("Fitting the classifier to the training set")

t0 = time()

param\_grid = {

    "C": loguniform(1e3, 1e5),

    "gamma": loguniform(1e-4, 1e-1),

}

clf = RandomizedSearchCV(

    SVC(kernel="rbf", class\_weight="balanced"), param\_grid, n\_iter=10

)

clf = clf.fit(X\_train\_pca, y\_train)

print("done in %0.3fs" % (time() - t0))

print("Best estimator found by grid search:")

print(clf.best\_estimator\_)

# %%

# Quantitative evaluation of the model quality on the test set

print("Predicting people's names on the test set")

t0 = time()

y\_pred = clf.predict(X\_test\_pca)

print("done in %0.3fs" % (time() - t0))

print(classification\_report(y\_test, y\_pred, target\_names=target\_names))

ConfusionMatrixDisplay.from\_estimator(

    clf, X\_test\_pca, y\_test, display\_labels=target\_names, xticks\_rotation="vertical"

)

plt.tight\_layout()

plt.show()

# %%

# Qualitative evaluation of the predictions using matplotlib

def plot\_gallery(images, titles, h, w, n\_row=3, n\_col=4):

    """Helper function to plot a gallery of portraits"""

    plt.figure(figsize=(1.8 \* n\_col, 2.4 \* n\_row))

    plt.subplots\_adjust(bottom=0, left=0.01, right=0.99, top=0.90, hspace=0.35)

    for i in range(n\_row \* n\_col):

        plt.subplot(n\_row, n\_col, i + 1)

        plt.imshow(images[i].reshape((h, w)), cmap=plt.cm.gray)

        plt.title(titles[i], size=12)

        plt.xticks(())

        plt.yticks(())

# %%

# plot the result of the prediction on a portion of the test set

def title(y\_pred, y\_test, target\_names, i):

    pred\_name = target\_names[y\_pred[i]].rsplit(" ", 1)[-1]

    true\_name = target\_names[y\_test[i]].rsplit(" ", 1)[-1]

    return "predicted: %s\ntrue:      %s" % (pred\_name, true\_name)

prediction\_titles = [

    title(y\_pred, y\_test, target\_names, i) for i in range(y\_pred.shape[0])

]

plot\_gallery(X\_test, prediction\_titles, h, w)

# %%

# plot the gallery of the most significative eigenfaces

eigenface\_titles = ["eigenface %d" % i for i in range(eigenfaces.shape[0])]

plot\_gallery(eigenfaces, eigenface\_titles, h, w)

plt.show()

# %%

# Face recognition problem would be much more effectively solved by training

# convolutional neural networks but this family of models is outside of the scope of

# the scikit-learn library. Interested readers should instead try to use pytorch or

# tensorflow to implement such models.

**得到的到特征分类结果如下 ：**

**n\_samples: 1288**

**n\_features: 1850**

**n\_classes: 7**

**Extracting the top 150 eigenfaces from 966 faces**

**done in 12.331s**

**Projecting the input data on the eigenfaces orthonormal basis**

**done in 0.007s**

**Fitting the classifier to the training set**

**done in 3.398s**

**Best estimator found by grid search:**

**SVC(C=7889.653874463754, class\_weight='balanced', gamma=0.0025346109980461185)**

**Predicting people's names on the test set**

**done in 0.043s**

**precision recall f1-score support**

**Ariel Sharon 0.73 0.62 0.67 13**

**Colin Powell 0.75 0.90 0.82 60**

**Donald Rumsfeld 0.75 0.67 0.71 27**

**George W Bush 0.91 0.95 0.93 146**

**Gerhard Schroeder 0.91 0.80 0.85 25**

**Hugo Chavez 0.90 0.60 0.72 15**

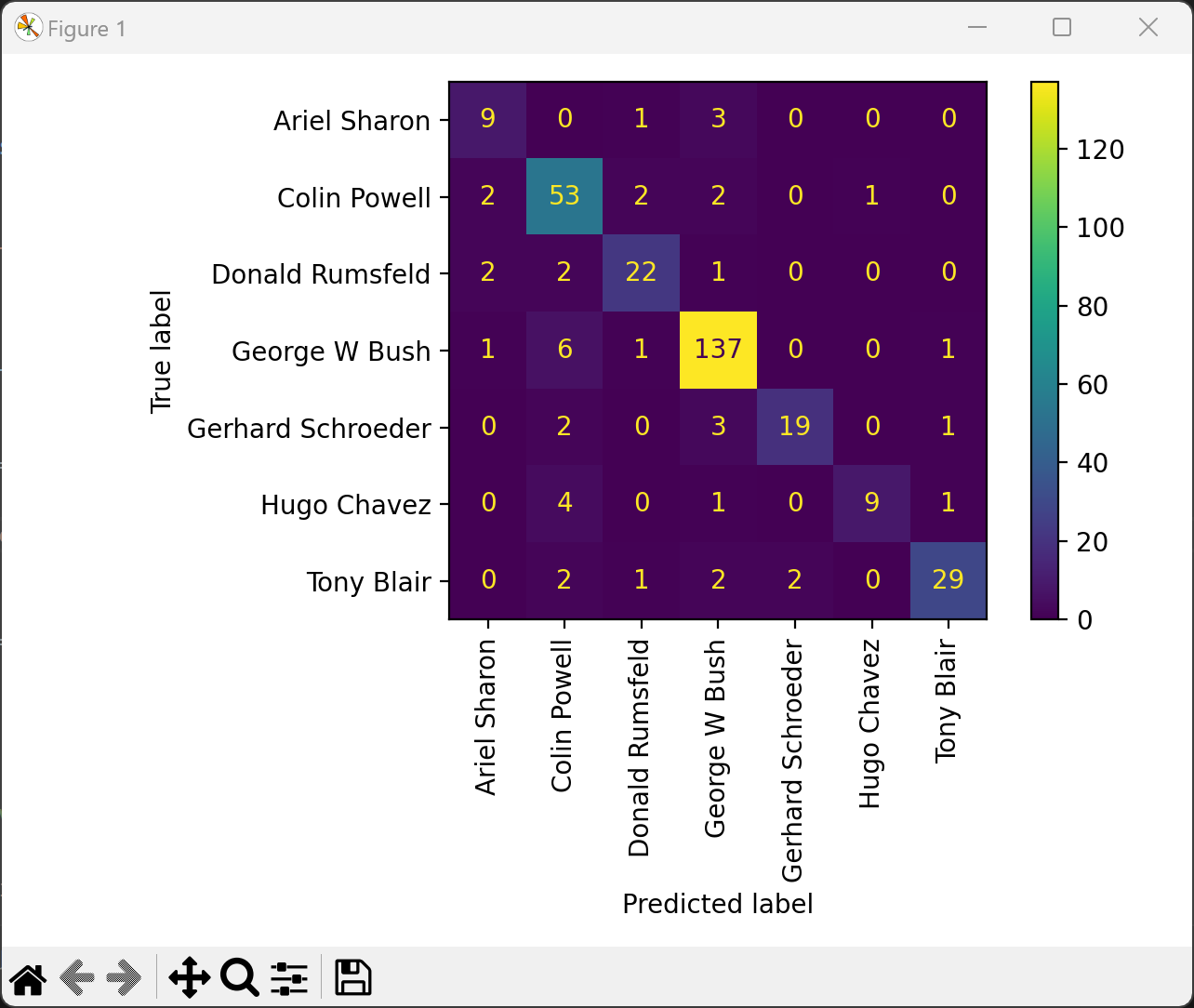
**Tony Blair 0.90 0.78 0.84 36**

**accuracy 0.86 322**

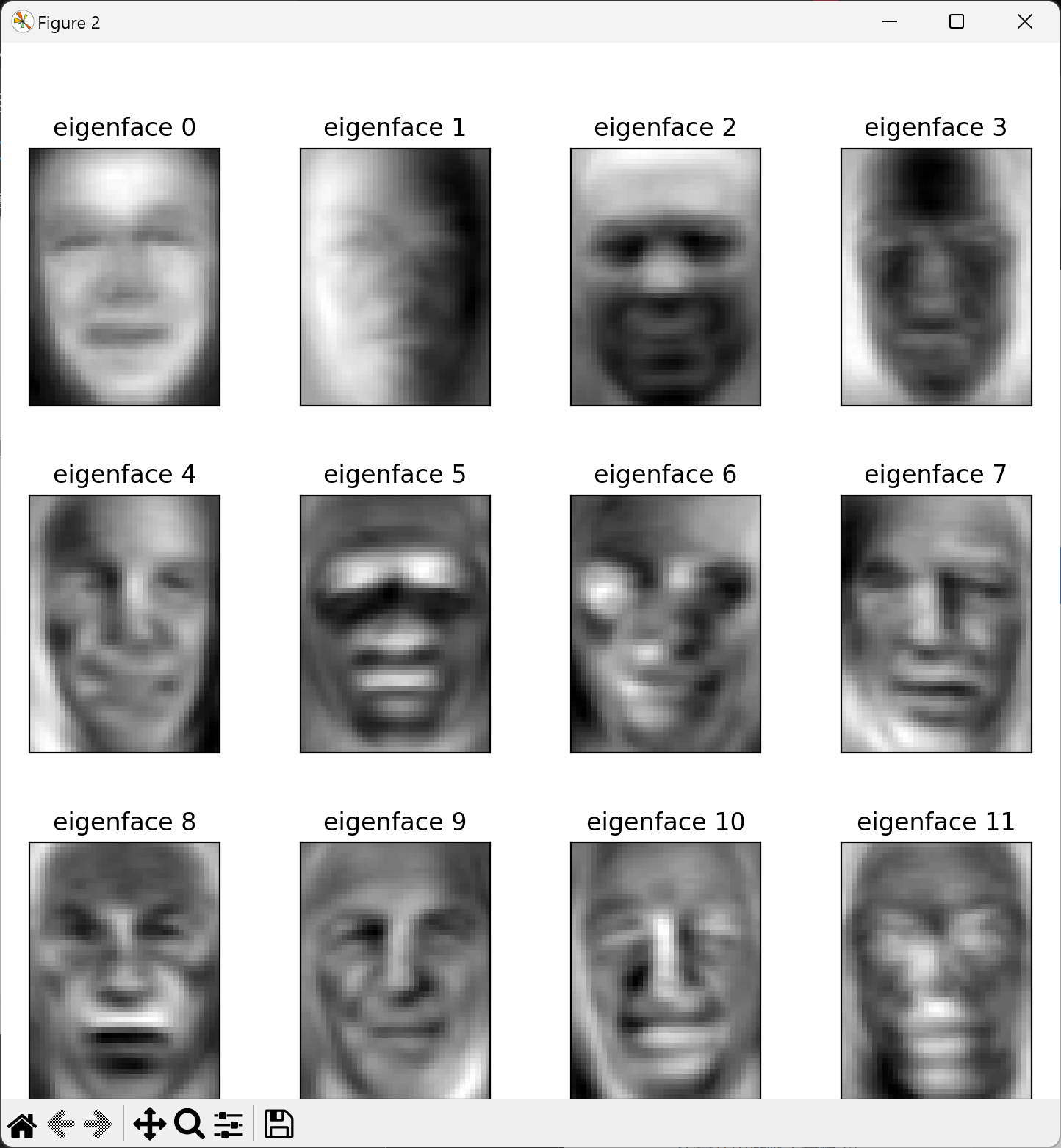
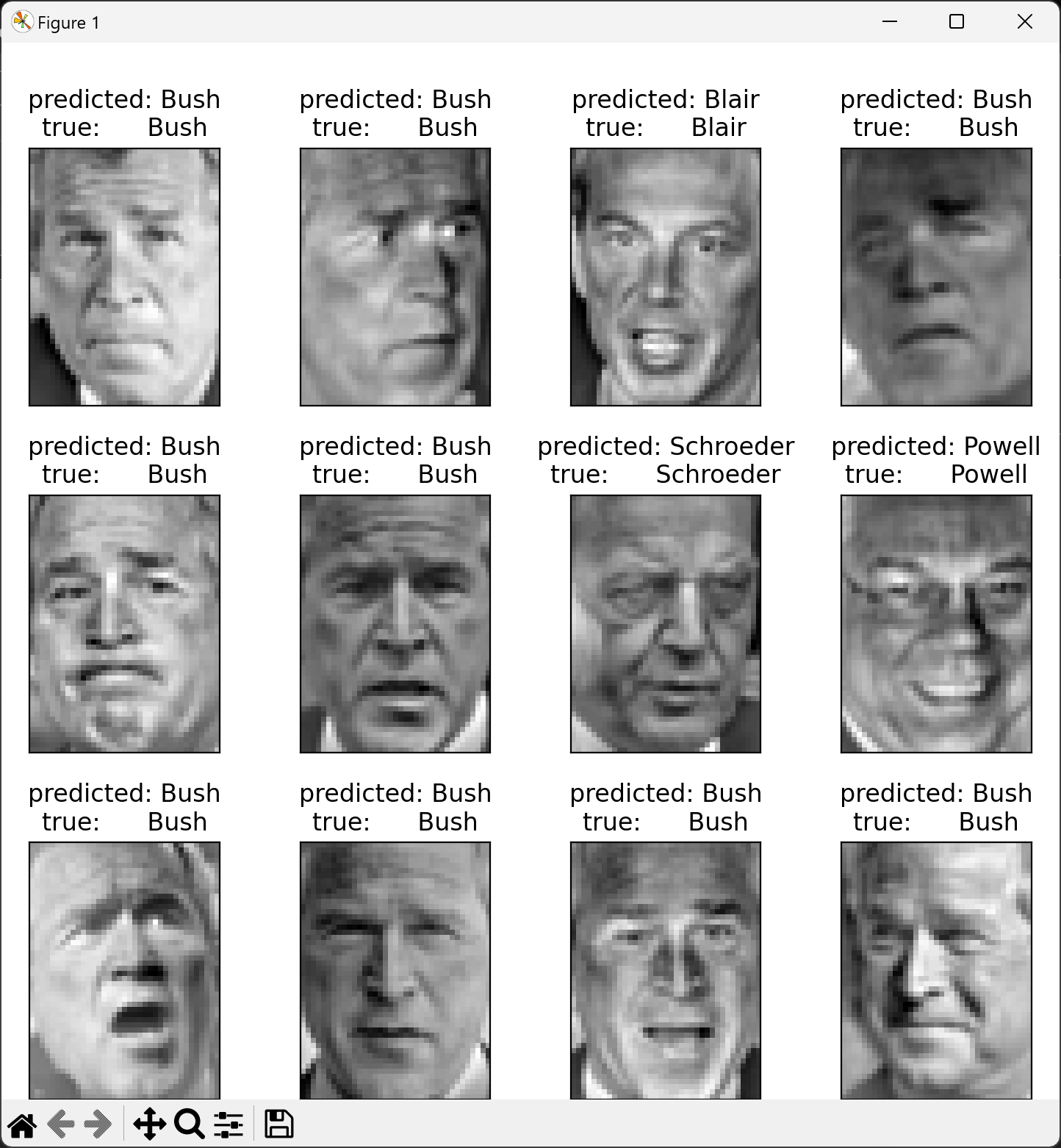
**macro avg 0.84 0.76 0.79 322**

**weighted avg 0.86 0.86 0.85 322**

**混淆矩阵如下**



**下图为预测结果和特征人脸展示：**



四、实验心得

在进行了KMeans聚类和轮廓分析的实验后，我有以下几点心得体会：

理论与实践的结合：通过这次实验，我更加深刻地理解了KMeans聚类算法的理论基础和实际操作过程。实验不仅仅是对理论的验证，更是对理论知识的深化和拓展。

参数选择的重要性：在KMeans聚类中，选择合适的聚类数（K值）对于结果至关重要。通过轮廓分析，我们可以更直观地评估不同K值对聚类效果的影响，这在实际应用中非常有用。

数据可视化的力量：轮廓图和散点图的绘制让我直观地看到了数据的分布和聚类结果，这种视觉化的方法对于理解数据和解释聚类效果非常有帮助。

算法评估的多维度：除了轮廓系数，Rand index和adjusted Rand index也是评估聚类效果的重要指标。它们从不同角度衡量聚类的质量，让我对聚类算法的评估有了更全面的认识。

实验操作的挑战：在实验过程中，我遇到了一些挑战，比如参数调整和结果解释。这些挑战让我意识到实际操作中需要灵活运用理论知识，并且要有耐心和细心去调试和优化。

创新思维的培养：在面对实验中的难题时，我学会了如何思考问题的本质，并尝试创新的方法来解决问题，这对于我的科研能力和创新思维的培养非常有益。

在本次学习过程中，我深入研究了传统机器学习框架下基于特征的人脸表示方法，并通过实践对相关技术和工具进行了详细了解。具体的收获如下：

学习传统机器学习框架基于特征的人脸表示方法

我学习了基于传统机器学习方法的人脸表示技术，特别是特征提取和表示方面的内容。传统的特征表示方法通常通过提取人脸图像中的几何特征、纹理特征等来描述人脸的外观。这些特征包括了面部关键点、边缘信息以及颜色分布等。通过这些特征，我们能够将人脸从高维空间映射到低维空间，从而降低计算复杂度，提高分类性能。

学习 scikit-learn 库中 PCA 相关模块的使用

在人脸识别过程中，主成分分析（PCA）是常用的降维技术。我通过学习 scikit-learn 库中的 PCA 模块，掌握了如何使用 PCA 对人脸数据进行降维和特征提取。通过对高维数据进行线性变换，PCA 能够找到最重要的特征维度，减少冗余信息。这种降维技术不仅帮助我降低了计算负担，还提高了模型的泛化能力。在实践中，我利用 PCA 进行了特征脸表示，将人脸图像投影到主成分空间中，从而提取出最具代表性的特征。

学习 scikit-learn 库中 SVM 相关模块的使用

我还学习了支持向量机（SVM）在分类任务中的应用，特别是在多类别分类问题中的使用。SVM 是一种强大的分类算法，通过构建最大间隔的超平面来区分不同类别的数据。在 scikit-learn 中，我掌握了如何利用 SVM 模块进行模型训练和分类，并通过对人脸数据的分类，进一步了解了 SVM 在图像分类中的优势和效果。

了解一些多类别分类评价指标

在多类别分类问题中，如何评估模型的性能至关重要。我学习了常见的分类评价指标，如准确率（Accuracy）、精确度（Precision）、召回率（Recall）、F1 分数等，并了解了它们在不同任务中的应用。通过这些指标的综合评价，可以更全面地了解模型的分类效果，尤其是在不均衡数据集中的表现。

总体收获

通过本次学习，我不仅掌握了传统机器学习框架下基于特征的人脸表示方法，还深入了解了 PCA 降维技术、SVM 分类算法以及多类别分类的评价指标。这些知识不仅帮助我在实际的人脸识别任务中提升了算法的性能，还让我在多类别分类问题中具备了更全面的评估能力。对于未来深入研究机器学习和计算机视觉技术打下了坚实的基础。