鸢尾花数据集的无监督学习分析

1. 数据预处理

数据预处理是数据分析的第一步，包括读取数据、处理缺失值、异常值等。数据的质量对分析结果有很大的影响，因此需要对数据进行初步的处理和检查，然后通过数据可视化，更好地理解数据的特征和分布。

针对数据预处理，可以按照以下步骤进行：

1．读取鸢尾花数据集：使用合适的库（如 pandas）读取数据集，转换为数据框格式。

df_iris=pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',names=['sepal length', 'sepal width', 'petal length', 'petal width', 'class'])

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

2．处理缺失值：检查数据集是否存在缺失值，如果存在，则需要对缺失值进行处理。可以选择直接删除含有缺失值的数据行，或者使用均值、中位数等方法进行填充。

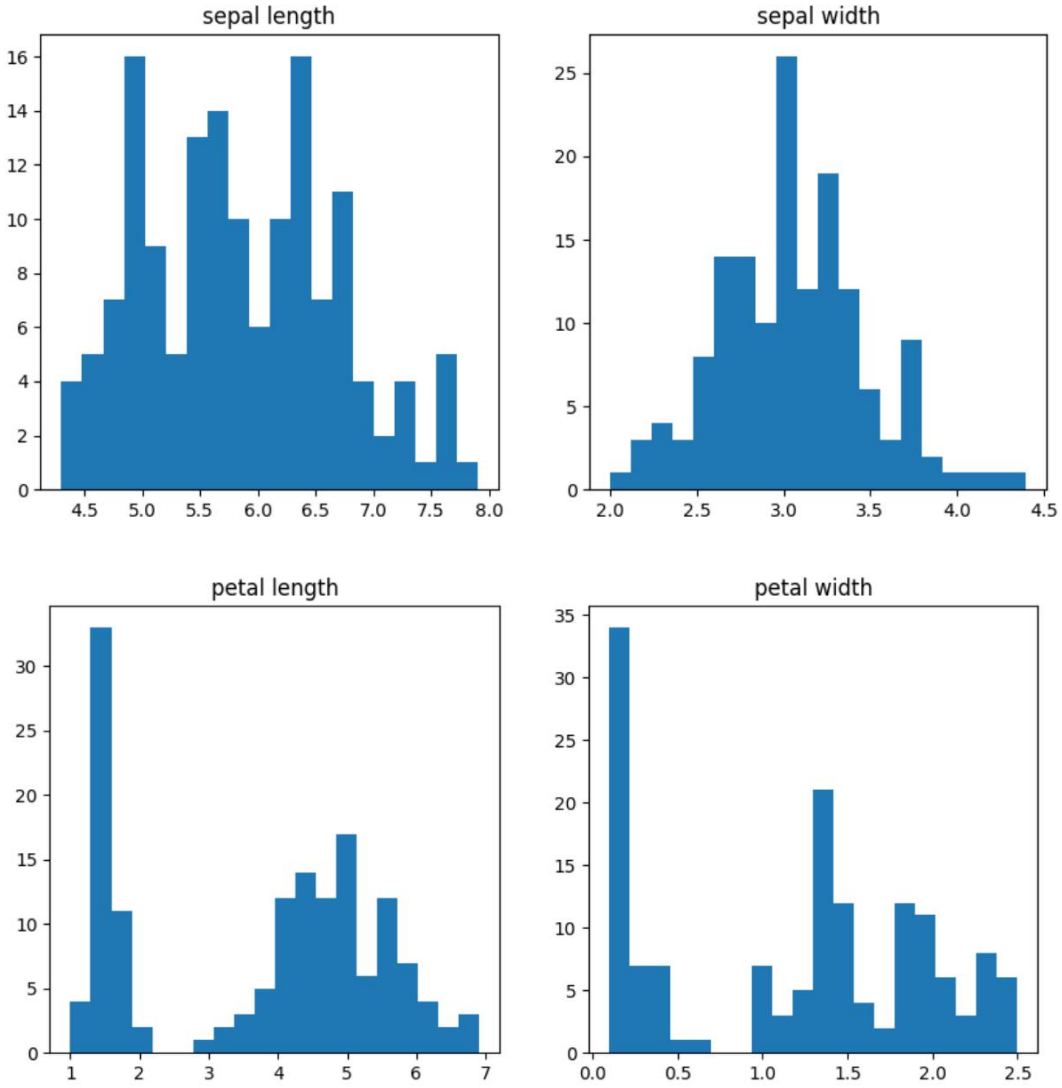
```
df_iris = df_iris.dropna()  # 删除数据集中所有包含缺失值的行
```

3．处理异常值：检查数据集是否存在异常值，如果存在，则需要对异常值进行处理。可以选择直接删除异常值所在的数据行，或者使用插值等方法进行填充。

```
df_iris = df_iris[(df_iris['sepal length'] > 0) & (df_iris['sepal width'] > 0)&
```

(df_iris['petal length'] > 0) & (df_iris['petal length'] > 0)]

4．数据可视化：使用合适的可视化工具（如 matplotlib）对数据进行可视化，观察数据的分布情况，是否存在异常点等信息。可以绘制散点图、箱线图等常见的可视化方式。



通过以上四个直方图的分析可以得出以下结论：

1．萼片长度（sepal length）和萼片宽度（sepal width）两个特征的分布大致呈正态分布，其中萼片长度的均值略高于萼片宽度的均值。

2．花瓣长度（petal length）和花瓣宽度（petal width）两个特征的分布明显不是正态分布，而是有两个峰值的分布。其中花瓣长度的峰值集中在 4-5 之间，花瓣宽度的峰值集中在 1-1.5 之间。

3．数据经过处理后，不存在缺失值和异常值。

以上步骤可以帮助本文准确、全面地了解数据集的特征和分布情况，为后续的聚类分析和降维分析打下基础。

2. 聚类分析

聚类分析是一种无监督学习方法，旨在将具有相似特征的对象归为一类，同时将不同类别的对象区分开来。聚类分析的目的是找出一组未知类别的样本数据的内在结构，这些样本数据可能具有相似的特征，但是又与其他样本数据不同。聚类分析可以在不了解样本数据标签的情况下，发现数据的内在规律和结构，帮助进行数据分析、特征提取和模式识别等任务。

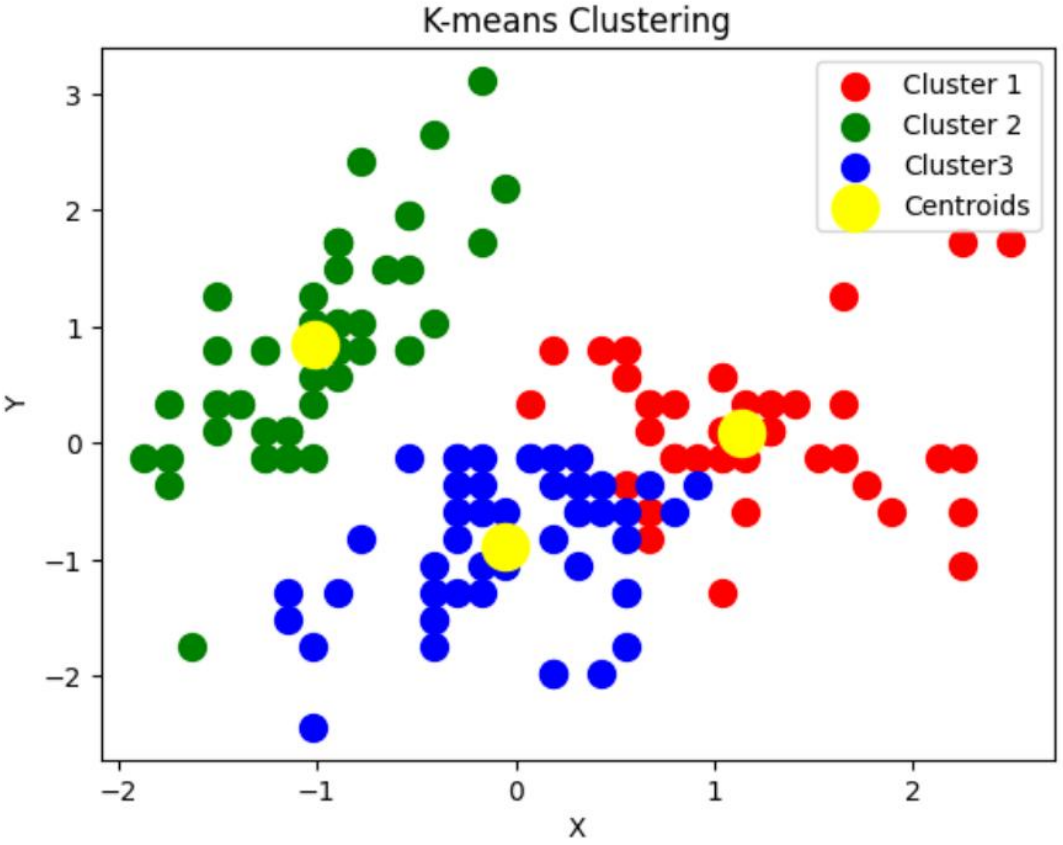
聚类分析通常使用距离或相似度来计算数据点之间的相似程度。聚类算法可以分为基于原型的聚类和层次聚类两种类型。基于原型的聚类算法将每个聚类表示为一个原型，比如聚类中心或是某个具有代表性的数据点。基于原型的聚类算法通常采用 K-Means 算法或 GMM（高斯混合模型）算法。层次聚类算法将数据点逐渐合并为越来越大的簇，可以分为自上而下的凝聚聚类和自下而上的分裂聚类。本文主要采用 K-means 方法和层次聚类方法。

2.1 K-means

K-means 是一种基于距离的聚类方法，它将数据点分为 k 个簇，使得同一簇内的数据点之间的距离最小，不同簇之间的距离最大。K-means 算法的基本思想是：首先随机选择 k 个中心点，然后将所有数据点分配到最近的中心点所在的簇，接着重新计算每个簇的中心点，重复上述过程直到收敛为止。具体代码如下：

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
# 对数据进行切片，获得特征数据
X = df_iris.iloc[:, :-1].values
lis = df_iris.loc[:, ['class']].values
y = [0 if x == 'Iris-setosa' else 1 if x == 'Iris-versicolor' else 2 for x in lis]
# 数据预处理：使得每个特征的均值为 0，标准差为 1
X_std = StandardScaler().fit_transform(X)
kmeans = KMeans(n_clusters=3,random_state=42)
y_kmeans = kmeans.fit_predict(X_std)
# 可视化聚类结果
plt.scatter(X_std[y_kmeans == 0, 0], X_std[y_kmeans == 0, 1],s=100, c='red',
label='Cluster 1')
plt.scatter(X_std[y_kmeans == 1, 0], X_std[y_kmeans == 1, 1],s=100, c='green',
```

```
label='Cluster 2')
    plt.scatter(X_std[y_kmeans == 2, 0], X_std[y_kmeans == 2, 1],s=100, c='blue', label='Cluster3')
    # 在散点图中绘制聚类中心点
    plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c='yellow', label='Centroids')
    plt.title('K-means Clustering')
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.legend()
    plt.show()
```

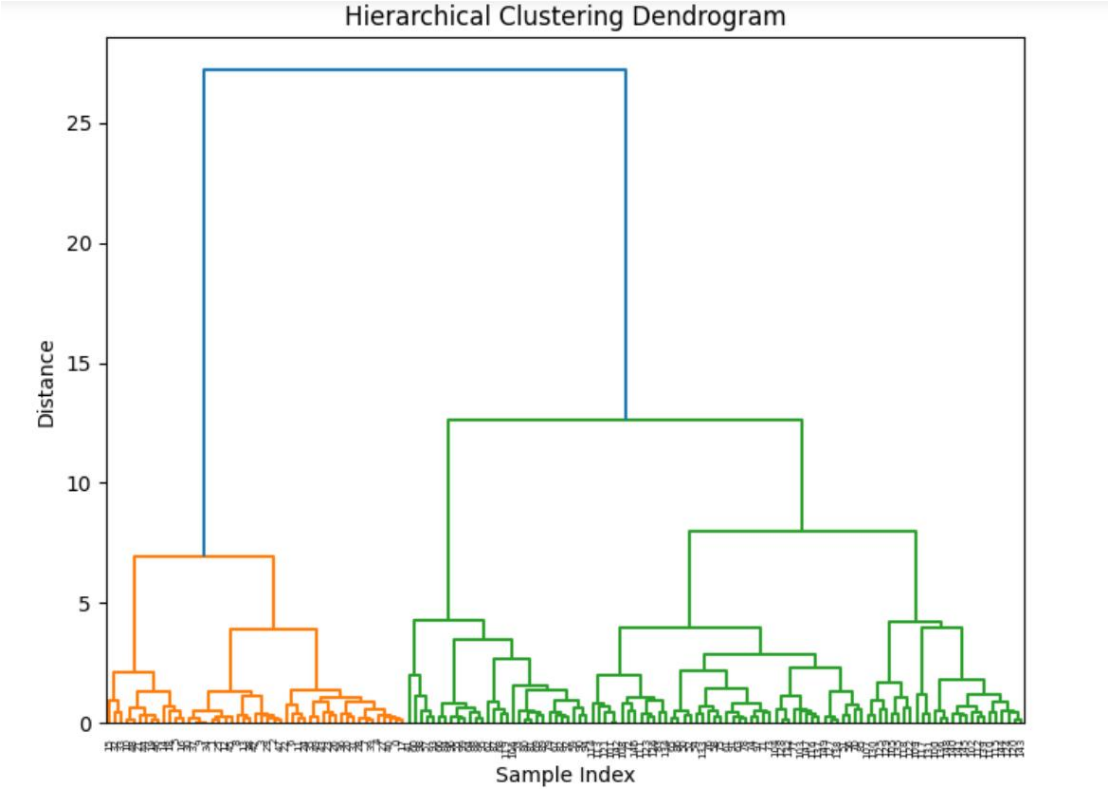


通过 K-means 聚类，本文将鸢尾花数据集中的数据分成了 3 个簇，每个簇中的数据点都具有相似的特征。本文还可以根据聚类中心点的位置，将数据点划分到离它们最近的簇中。在可视化结果中，不同的颜色代表不同的簇，而黄色的点则表示聚类的中心点。可以看出，聚类结果相对比较明显，不同簇之间的边界比较清晰，聚类结果与真实类别有一定程度上的相似性，但并不完全匹配，这表明使用 K-means 算法对鸢尾花数据集进行聚类是一个较好的选择。

2.2 层次聚类

层次聚类算法是一种自底向上的聚类算法，它将每个样本看作一个单独的簇，然后逐渐将它们合并成更大的簇，直到所有的样本都被合并到一个簇中。层次聚类算法可以用于生成一个树形图，称为树状图，以可视化聚类结果。具体代码如下：

```
from scipy.cluster.hierarchy import dendrogram,linkage
#  使用层次聚类算法进行聚类分析
# ward 方法：以最小化方差为优化目标来进行层次聚类，即将两个簇合并时，它们内部点与簇中心的平方距离之和的变化量
Z = linkage(X_std,method='ward')
#  可视化树状图
plt.figure(figsize=(8,6))
dendrogram(Z)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Sample Index')
plt.ylabel('Distance')
plt.show()
```



以上代码使用了 Scipy 库中的层次聚类方法（hierarchy），并使用 ward 方法进行聚类。聚类结果的树状图通过 dendrogram 函数绘制出来，其中横轴表示数

据样本的索引，纵轴表示样本之间的距离。

从可视化结果中可以看到，树状图从下往上的分支越来越少，这是由于层次聚类算法从下往上合并相似的数据点所导致的。通过层次聚类算法生成的树状图，本文可以观察到不同样本之间的相似度以及它们被分配到不同的聚类中的情况。树状图中的每个叶子节点代表一个样本，叶子节点之间的距离表示样本之间的差异程度，距离越近代表两个样本的相似度越高，反之则越低。同时，本文还可以观察到树状图的结构，不同的高度表示了聚类的数量以及聚类之间的差异程度。

2.3 对比

K-means 和层次聚类方法各有优劣。

K-means 的优点：相对简单且易于理解和实现。计算速度较快，适用于大型数据集。可以使用不同的距离度量方法和簇数来进行调整。

K-means 的缺点：对于非凸形状或不同密度的簇效果不佳。对于数据分布不平衡或噪声敏感，需要进行数据预处理和异常值处理。

层次聚类的优点：通过可视化树状图可以更直观地了解簇之间的关系。不需要预先指定簇数，可以根据需要确定不同的簇数。对于非凸形状或不同密度的簇效果较好。

层次聚类的缺点：计算复杂度高，不适用于大型数据集。受数据噪声和异常值的影响较大。对于大型数据集，需要进行采样或者其他优化方法。

针对鸢尾花数据集进行 K-means 聚类和层次聚类的结果对比如下：

在 K-means 聚类的结果中，本文选择了 3 个聚类中心，可以看到得到了 3 个不同的聚类。其中 Cluster 1 代表花萼长度和花萼宽度相对较小的一类花，Cluster 2 代表花瓣长度和花瓣宽度相对较大的一类花，Cluster 3 代表花萼长度和花萼宽度相对较大的一类花。在图像中，本文还标出了每个聚类的中心点（黄色的点）。

而在层次聚类的结果中，本文选择了 Ward 方法进行聚类，可以看到通过树状图展示了聚类的过程和结果。从树状图中可以看出，当聚类数为 3 时，本文得到了 3 个不同的聚类。在图像中，本文还展示了每个样本所属的聚类，可以看出每个聚类的特点与 K-means 的结果类似。

相对于 K-means 聚类，层次聚类更加直观，可以通过树状图展示聚类的过程和结果，更容易理解和解释。而 K-means 聚类则更适合大规模数据的聚类，并且运算速度更快。因此，两种聚类方法在不同的数据场景下，具有不同的优缺点。在鸢尾花数据集中，这两种方法都可以得到类似的聚类结果，但对于不同的数据集，选择不同的聚类算法可能会得到更好的结果。

3. 降维分析

降维分析是指将高维数据映射到低维空间中，保留数据主要特征的同时减少数据维度，以便于可视化、分析和处理高维数据。在机器学习和数据挖掘领域中使用降维分析来减少特征数量可以提高模型的效率和准确性。

降维分析可以分为两种主要的方法：线性降维和非线性降维。线性降维方法包括主成分分析（PCA）和线性判别分析（LDA），它们可以保留数据的主要信息，但无法捕捉数据中的非线性结构。而非线性降维方法如多维尺度分析（MDS）和局部线性嵌入（LLE）则可以捕捉数据中的非线性结构，但在计算复杂度和处理大规模数据时存在一定的限制。

降维分析的优点是可以将高维数据可视化展示，便于理解和分析数据的特征，同时可以减少特征数量，降低模型复杂度和计算复杂度，提高模型效率和准确性。缺点是降维过程可能丢失部分数据信息，需要根据具体应用场景权衡利弊。本文主要采用线性降维方法（PCA 和 LDA）对鸢尾花数据集进行降维。

3.1 主成分分析（PCA）

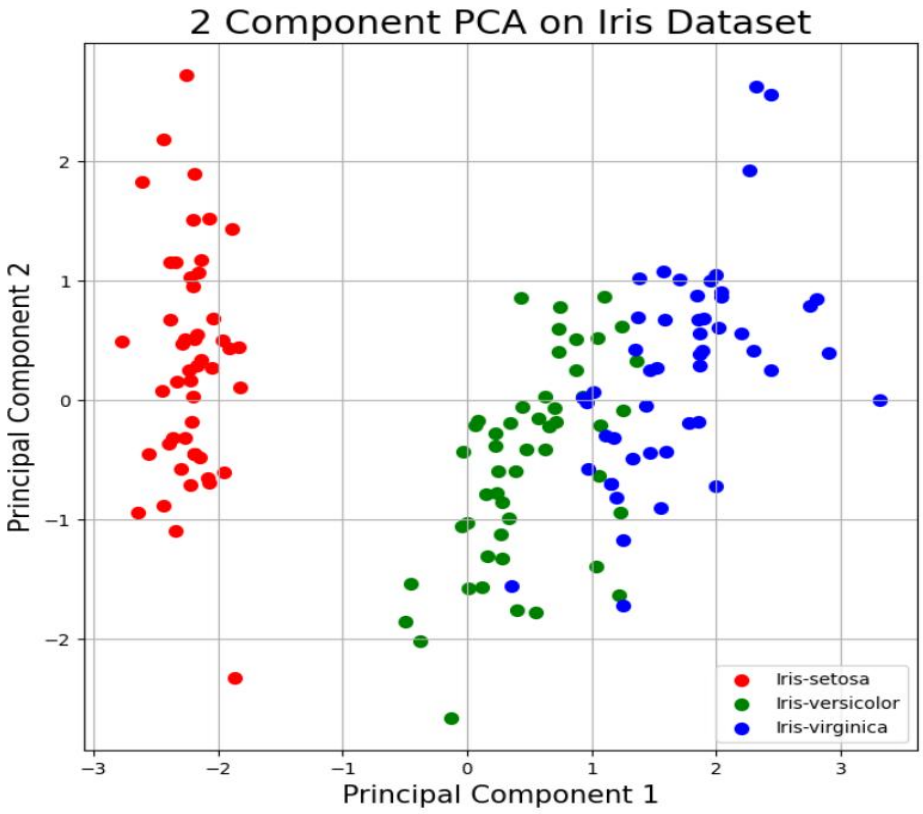
主成分分析（Principal Component Analysis，PCA）是一种常见的降维方法，用于将高维数据降至低维，同时保留尽可能多的原始信息。

PCA 的主要思想是将原始数据转换为一组新的变量，这些新变量是原始变量的线性组合，且彼此独立。这些新变量按照方差的大小排序，方差越大的新变量保留的原始信息越多。因此，可以只保留前面几个主成分，而忽略方差较小的主成分，从而实现降维的目的。具体代码如下：

```
from sklearn.decomposition import PCA
# 创建 PCA 对象
pca_two = PCA(n_components=2)
# 对数据集进行 PCA 降维
principalComponents = pca_two.fit_transform(X_std)
# 将降维后的数据转换为 DataFrame
principalDf=pd.DataFrame(data=principalComponents,columns=['principal component 1', 'principal component 2'])
# 将降维后的数据与标签合并
```

```
finalDf = pd.concat([principalDf, df_iris[['class']]], axis=1)

# 可视化 PCA 降维后的数据
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(1, 1, 1)
ax.set_xlabel('Principal Component 1', fontsize=15)
ax.set_ylabel('Principal Component 2', fontsize=15)
ax.set_title('2 Component PCA on Iris Dataset', fontsize=20)
targets = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
colors = ['r', 'g', 'b']
for target, color in zip(targets, colors):
    indicesToKeep = finalDf['class'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1'],
               finalDf.loc[indicesToKeep, 'principal component 2'], c=color,
s=50)
ax.legend(targets)  # 在图例中添加类别名称
ax.grid()  # 给图像添加网格线
plt.show()
```



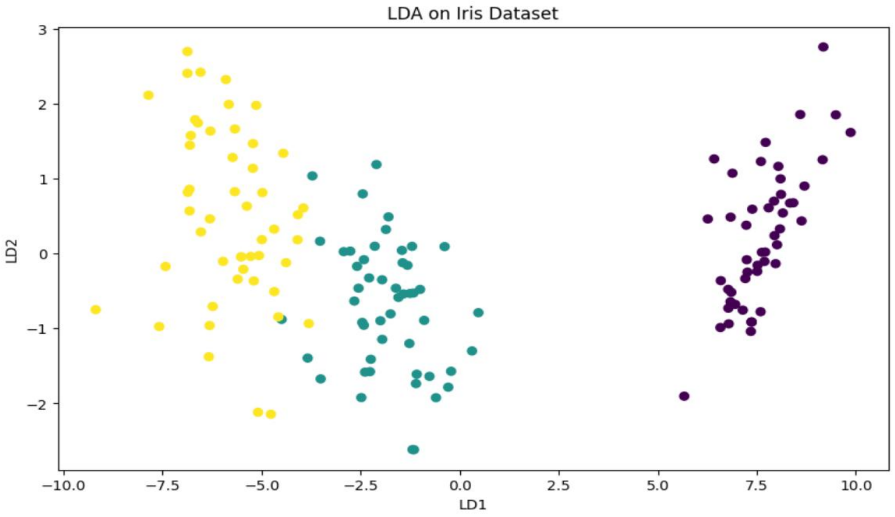
通过可视化图形可以看出，经过 PCA 降维后，原始数据集的四个特征被压

缩到了两个主成分上，可以看到这两个主成分可以很好地将三种不同的鸢尾花区分开来。其中，红色点代表 Iris-setosa，绿色点代表 Iris-versicolor，蓝色点代表 Iris-virginica。可以看到，这三个类别被明显地分离在不同的区域，这说明 PCA 降维后的数据可以很好地保留原始数据的信息，并且有助于更好地理解数据集的结构。

3.2 线性判别分析（LDA）

线性判别分析（Linear Discriminant Analysis，LDA）是一种经典的降维算法，它既可以作为数据降维的手段，也可以作为分类算法。其基本思想是将原始数据投影到一个低维度的线性空间中，使得同一类样本的投影点尽量接近，不同类样本的投影点尽量远离。与主成分分析（PCA）不同的是，LDA 是有监督的降维方法，需要先知道每个样本所属的类别。通过计算类别内散度矩阵和类别间散度矩阵，可以得到一个判别函数，利用这个判别函数可以将新的样本进行分类。具体代码如下：

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA(n_components=2)
X_lda = lda.fit_transform(X_std, y)
plt.figure(figsize=(10, 6))
plt.scatter(X_lda[:, 0], X_lda[:, 1], c=y, cmap='viridis')
plt.xlabel('LD1')
plt.ylabel('LD2')
# plt.legend()
plt.title('LDA on Iris Dataset')
plt.show()
```



从上述代码的可视化图形可以看出，使用线性判别分析(LDA)对鸢尾花数据集进行降维后，我们可以看到三个品种的鸢尾花在二维空间中被比较好地区分开来。这表明 LDA 方法对于该数据集的分类问题是有效的。其中，LD1 和 LD2 分别是第一个和第二个线性判别变量，它们被用来最大程度地区分不同品种的鸢尾花。

3.3 对比

PCA 和 LDA 都是降维技术，它们可以用于从高维数据中提取有用的特征并将其投影到较低维的空间中，从而减少数据的维度。然而，两者的目的和方法略有不同，因此它们在特征提取方面的结果也略有不同。

PCA 是一种无监督的方法，其目标是找到可以最大程度保留原始数据方差的新特征，这些新特征是原始特征的线性组合。这些新特征称为主成分，每个主成分都与原始数据的一个方向相关。PCA 通常用于降维，以便可以更轻松地可视化数据，并且可以用于数据压缩和去除噪声。

LDA 是一种有监督的方法，它的目标是找到可以最大程度区分不同类别的新特征，这些新特征是原始特征的线性组合。与 PCA 不同，LDA 在降维的同时尽可能地保留类别信息，从而提高分类准确率。它通常用于特征提取，以便可以更好地分类和识别不同的对象。

对于鸢尾花数据集，使用 PCA 和 LDA 降维后，可以从二维平面上观察不同类别之间的分布情况。从可视化结果来看，PCA 降维后的数据集没有很好的区分不同的类别，而 LDA 降维后的数据集则能够清晰地区分出不同的鸢尾花品种。这是由于 LDA 能够最大程度地提取有区分性的特征，而 PCA 则只是尽可能地保留了原始数据的方差，但没有考虑不同类别之间的差异。因此，在需要对不同类别进行区分的任务中，LDA 通常比 PCA 更有效。

4. 实验对比

在进行聚类分析和降维分析时，不同的方法会对分析结果产生不同的影响。在选择方法时，需要考虑方法的优缺点以及数据的特点，从而选择最适合的方法。

对于聚类方法，K-means 和层次聚类都是常用的方法。K-means 速度快，但需要手动指定簇数；层次聚类不需要指定簇数，但计算复杂度较高。在鸢尾花数据集上，K-means 和层次聚类都能得到比较好的聚类结果，但 K-means 需要事先指定簇数，而层次聚类能够自适应选择簇数，因此在数据集较大时，层次聚类更

加适用。

对于降维方法，PCA 和 LDA 都是常用的方法。PCA 主要用于无监督降维，能够保留数据集中的大部分方差信息，但对于分类问题效果并不好；LDA 主要用于有监督降维，能够最大化类别之间的距离和最小化类别内部的距离，能够保留分类信息。在鸢尾花数据集上，PCA 和 LDA 都能够得到较好的降维效果，但 LDA 能够保留分类信息，因此更适合用于鸢尾花数据集这种分类问题。

因此，在选择聚类方法和降维方法时，需要考虑数据集的特点和问题的要求，选择最适合的方法。

5. 结果解释

经过聚类和降维分析，我们可以得出以下结论：

1．聚类分析：通过 K-means 聚类算法和层次聚类算法对鸢尾花数据集进行聚类分析，可以发现该数据集中存在着三个明显的类别，分别对应着三种不同的鸢尾花。

2．降维分析：通过 PCA 和 LDA 对鸢尾花数据集进行降维分析，可以得到二维特征空间中的数据分布情况。在 PCA 降维中可以看到不同种类的鸢尾花被分布在不同的区域，但是存在一定的重叠。而在 LDA 降维中可以看到不同种类的鸢尾花被明显地分开，说明 LDA 在特征提取上比 PCA 更加有效。

综上所述，聚类和降维分析都可以帮助我们了解数据集中的潜在特征和模式，从而更好地进行数据分析和建模。在选择聚类算法和降维算法时，需要根据具体的数据集特点和分析目的进行选择，并对结果进行解释和评估。