无监督学习 实验报告

PB17050965 朱子琦

实验环境

- python 3.8
- numpy 1.18.4
- matplotlib 3.2.2

实验内容

数据预处理与结果评测

数据预处理

原始数据各组数据尺度差异很大，很难正确判断是哪一组数据在其中产生影响，因此，先将其按比例缩放到[0,1]范围内，而后再按其到0.5的距离进行离散化，使得其能够分布在[-1,1]范围内

```
def data_preprocess(data):
    n, m = data.shape

    for i in range(1, m):
        max_value = data[:, i].max()
        min_value = data[:, i].min()
        data[:, i] = ((data[:, i] - min_value) /
                      (max_value - min_value) - 0.5) * 2

    label = data[:, 0]
    data = data[:, 1:]
    # print(label)
    # print(data)
    return data, label
```

评价函数

兰德系数

根据兰德系数公式统计计算

```python
def evaluate_RI(C, K):
    """
    evaluate_RI [使用兰德系数评价结果]

    Args:
        C ([dict]): [真实结果]
        K ([dict]): [聚类结果]

    Returns:
        [num]: [兰德系数值]
    """
    a, b, c, d = 0, 0, 0, 0

    for i in range(len(C)):
        for j in range(len(C)):
            if i == j:
                continue
            else:
                if C[i] == C[j] and K[i] == K[j]:
                    a += 1
                elif C[i] == C[j] and K[i] != K[j]:
                    b += 1
                elif C[i] != C[j] and K[i] == K[j]:
                    c += 1
                else:
                    d += 1
    RI = (a + d) / (a + b + c + d)
    return RI
```

轮廓系数

在进行轮廓系数计算时，需要确定数据的分类方式，而后根据分类去计算该点到周围各簇以及本簇内其他点的距离，最后根据得到的a(i) 和 b(i) 计算对应的SI

```python
def evaluate_SI(k,label, data, center):
    def distance(v1, v2):
        return (v2.T @ v2 - 2 * v1.T @ v2 + v1.T @ v1) ** 0.5

    n,m = data.shape
    board = np.zeros([n,n])

    for i in range(n):
        for j in range(n):
            board[i,j] = distance(data[i],data[j])
    def a(i):
        avr,count = 0,0
        for j in range(n):
            if j != i and label[i] == label[j]:
                avr += board[i,j]
                count += 1
        return avr/count
    def b(i):
        d = np.zeros([k])
        count = np.zeros([k])
        for j in range(n):
            d[label[j]] += board[i,j]
            count[label[j]] += 1
        for j in range(k):
            d[j] = d[j] / count[j]
        mind = np.inf
        for j in range(k):
            if j != label[i] and d[j] < mind:
                mind = d[j]
        return mind
    def s(i):
        ai,bi = a(i),b(i)
        return (bi-ai)/max(ai,bi)

    slist = []
    for i in range(n):
        slist.append(s(i))
    return sum(slist)/n
```

PCA

PCA 是将n维特征映射到k维上，这k维是全新的正交特征也被称为主成分，是在原有n维特征的基础上重新构造出来的k维特征。

对原始数据进行降维处理的过程中，要降低到k维需要得到最大的k个特征向量，通过np.linalg求对应矩阵特征向量，而后根据threhold选出前k个特征向量，用对应的特征向量进行线性变换，得到的矩阵就是降维后的矩阵，并通过函数返回值返回。

```python
def PCA(X, threhold=0.8):
    """
    PCA [使用PCA算法对数据进行降维处理]

    Args:
        X ([np.array([n,m])]): [待降维的数据集np.array([n,m])]]
        threhold (float, optional): [PCA 降维阈值，选取特征累加贡献率达到该阈值的前k个特征]. Defaul

    Returns:
        [np.array([n,k])]: [返回降低至k维后的数据集]
    """
    if threhold < 0 or threhold > 1:
        print("threhold should be between 0 and 1")
        return X

    # 计算输入数据矩阵的特征向量
    n, m = X.shape
    scatter_matrix = X.T @ X
    eig_val, eig_vec = np.linalg.eig(scatter_matrix)
    eig_pairs = [(np.abs(eig_val[i]), eig_vec[:, i]) for i in range(m)]
    eig_pairs.sort(reverse=True)

    # 计算需要达到threhold的k
    base,top,k  = sum(eig_val),0,m

    for i in range(m):
        top += eig_pairs[i][0]
        if top/base > threhold:
            k = i + 1
            break

    # 选取前k个特征向量
    feature = np.array([ele[1] for ele in eig_pairs[:k]])
    data = X @ feature.T
    return data
```

K-Means

k-means算法是通过创建k个中心，让数据点选择最近的中心称为簇。每轮迭代更新中心为当前簇的重心，而后进行下一轮迭代，直到每个点所在的簇不再发生改变则停止。

```python
def k_means(k, data):
    def distance(v1, v2):
        return (v2.T @ v2 - 2 * v1.T @ v2 + v1.T @ v1) ** 0.5

    def avrcenter(v1, v2, k1, k2):
        return (v1 + v2) / (k1 + k2)

    n, m = data.shape
    center = np.zeros([k, m])
    num = np.zeros([k])
    label = [1]
    next_label = []
    for i in range(k):
        center[i] = data[int(n / k * i)]
    t = 0
    while next_label != label:
        label = next_label.copy()
        next_label = [None]*n

        for i in range(n):
            d = []
            for j in range(k):
                d.append((distance(center[j], data[i]), j))
            d.sort()
            cluster = d[0][1]
            next_label[i] = cluster
            num[cluster] += 1
        for i in range(k):
            center[i] = np.zeros(m)
        for i in range(n):
            center[next_label[i]] += data[i]
        for i in range(k):
            center[i] = center[i] / num[i]
        num = np.zeros([k])
        t += 1

    return k,label,center
```

实验结果

PCA降维的threhold

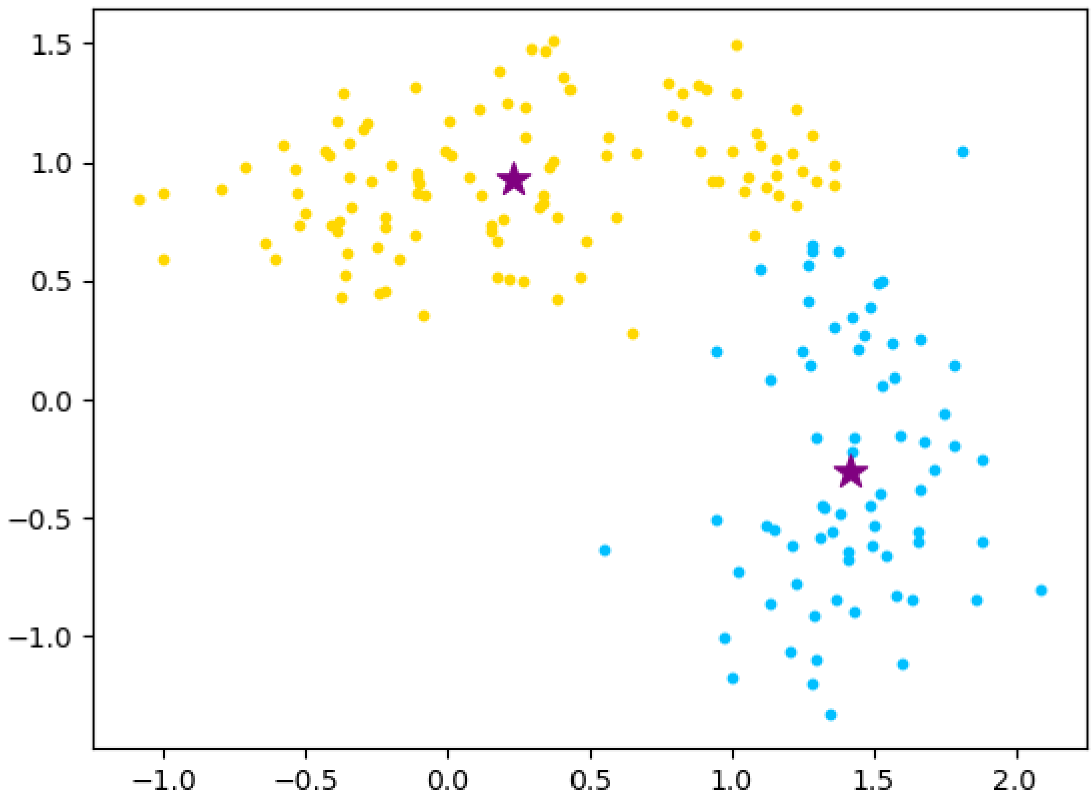
经实验，不同threhold范围内对应的结果维数k,兰德系数RI和轮廓系数SI为

threhold	k	兰德系数RI	轮廓系数SI
0.3759616388123276	1	0.7056433695169174	0.6316664749817953

threhold	k	兰德系数RI	轮廓系数SI
0.6261045429970125	2	0.8362216720624643	0.5199313736871968
0.7291660883240724	3	0.9191899955564019	0.505556006687245
0.785168030660275	4	0.9318225099980956	0.4407097834539247
0.838108938465557	5	0.9191899955564019	0.4007694696269683
0.8793872709770619	6	0.9191899955564019	0.36940825202038036
0.9140906731626023	7	0.9264901923443154	0.3475953691837858
0.9210998493733723	8	0.9348695486573986	0.3376958556676636
0.9308947405600075	9	0.9348695486573986	0.3266843483029067
0.9497886943727526	10	0.9348695486573986	0.3171869407121694
0.9677133169251664	11	0.9348695486573986	0.3038264436279519
0.9846275862386208	12	0.9348695486573986	0.3038264436279519
1.0	13	0.9348695486573986	0.3008938518500136

进行PCA处理

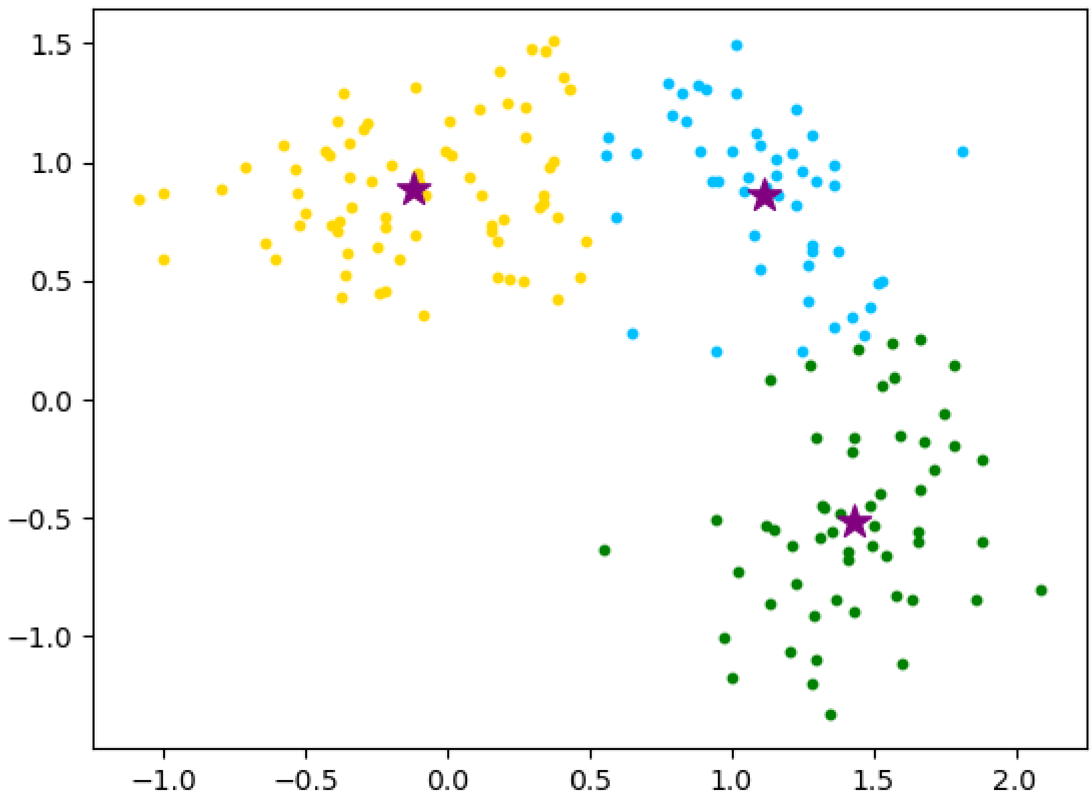
为便于可视化，使用PCA降低至二维
经过PCA后分为 2 个簇结果如图



Rand Index:0.6810131403542182

Silhouette Coefficient:0.5405649074932503

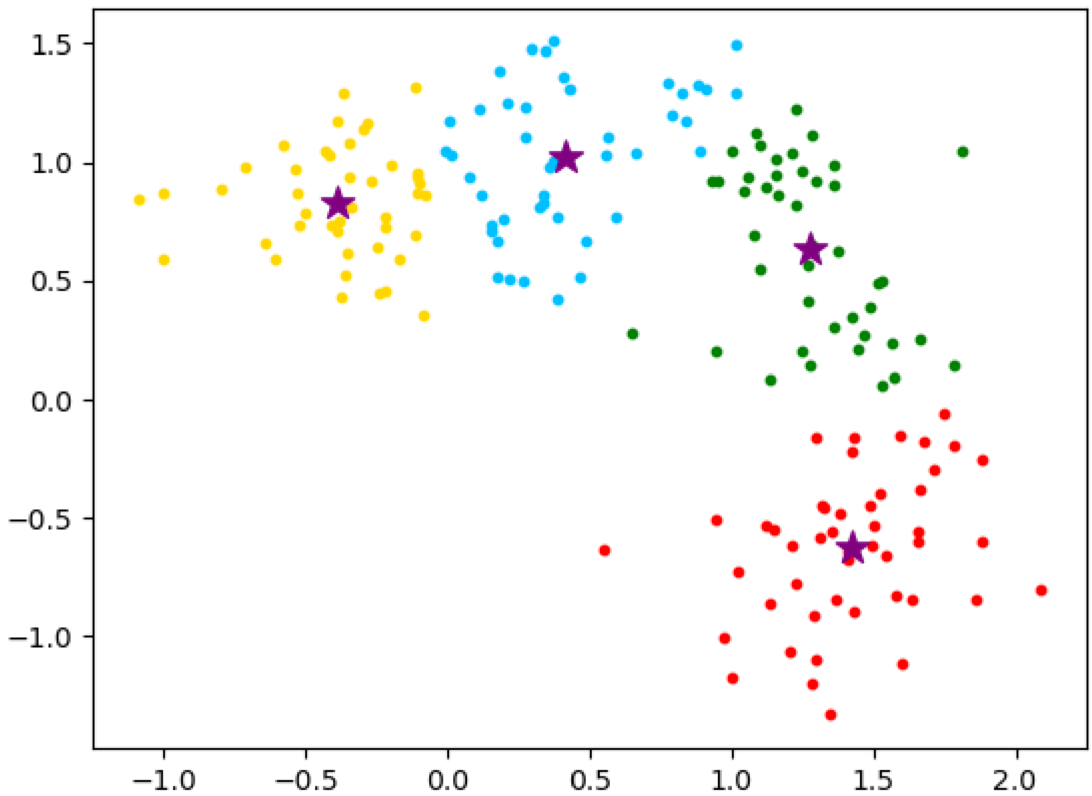
3 个簇结果如图



Rand Index:0.8362216720624643

Silhouette Coefficient:0.5199313736871968

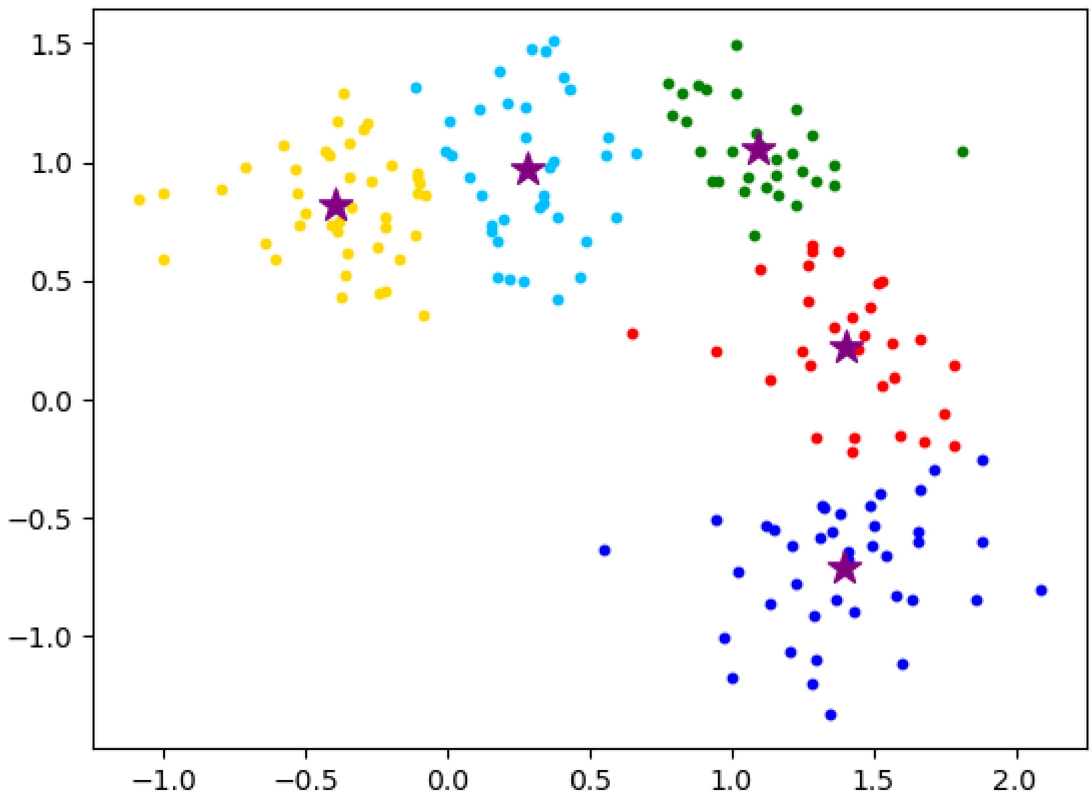
4个簇结果如图



Rand Index:0.816035040944582

Silhouette Coefficient:0.4488521926995579

5个簇结果如图



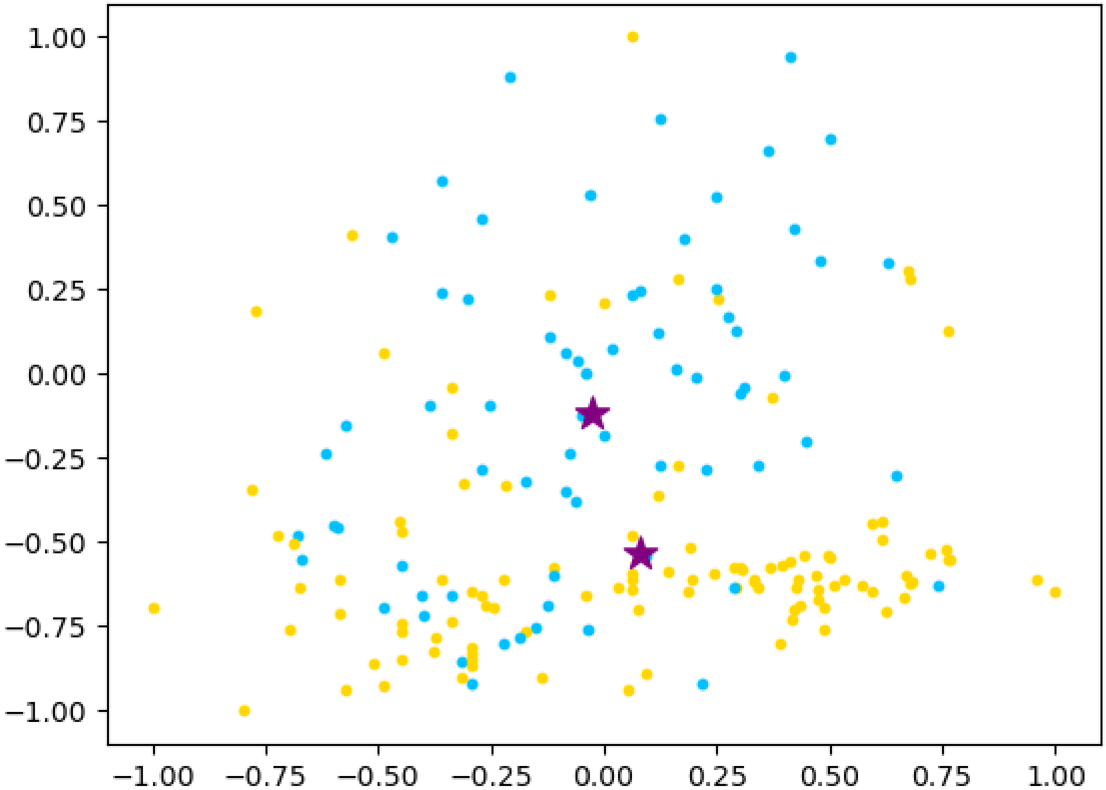
Rand Index:0.7769948581222624
Silhouette Coefficient:0.43815734585649574

不同簇数下的Si和Ri

k	兰德系数RI	轮廓系数SI
2	0.6810131403542182	0.5405649074932503
3	0.8362216720624643	0.5199313736871968
4	0.816035040944582	0.4488521926995579
5	0.7769948581222624	0.43815734585649574

不进行PCA处理

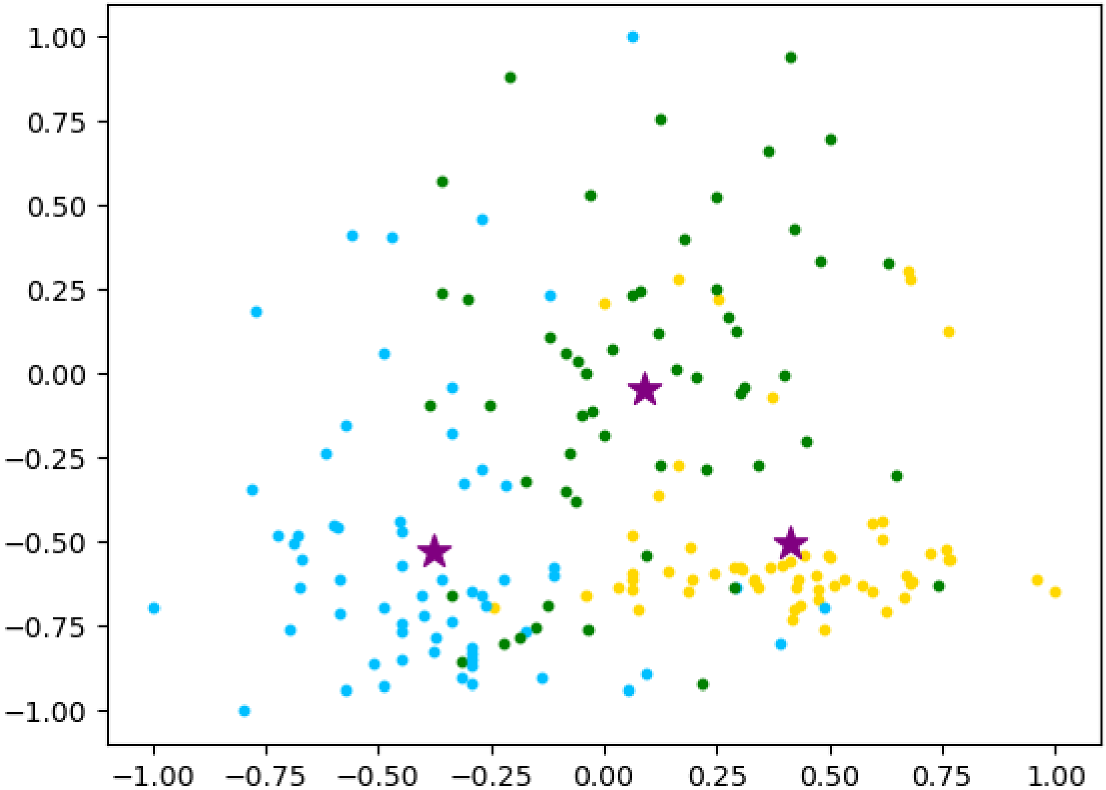
2 个簇结果如图



Rand Index:0.6810131403542182

Silhouette Coefficient:0.2987221815974774

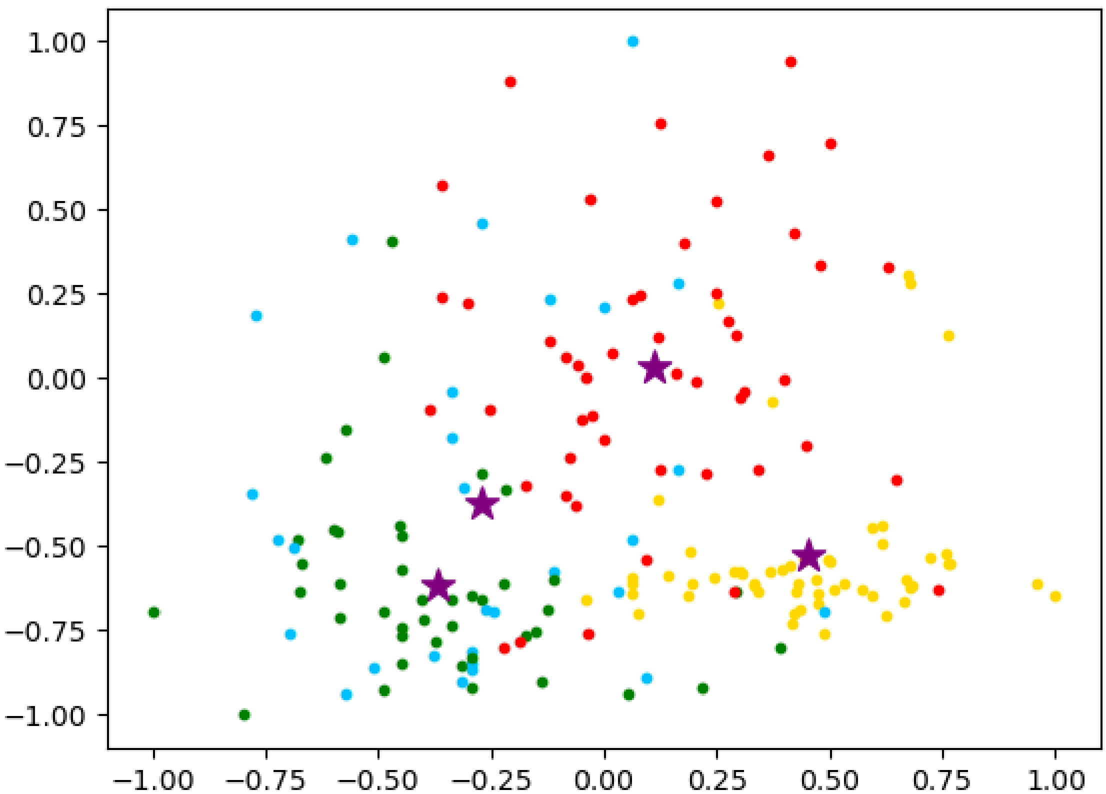
3个簇结果如图



Rand Index:0.9348695486573986

Silhouette Coefficient:0.30089385185001355

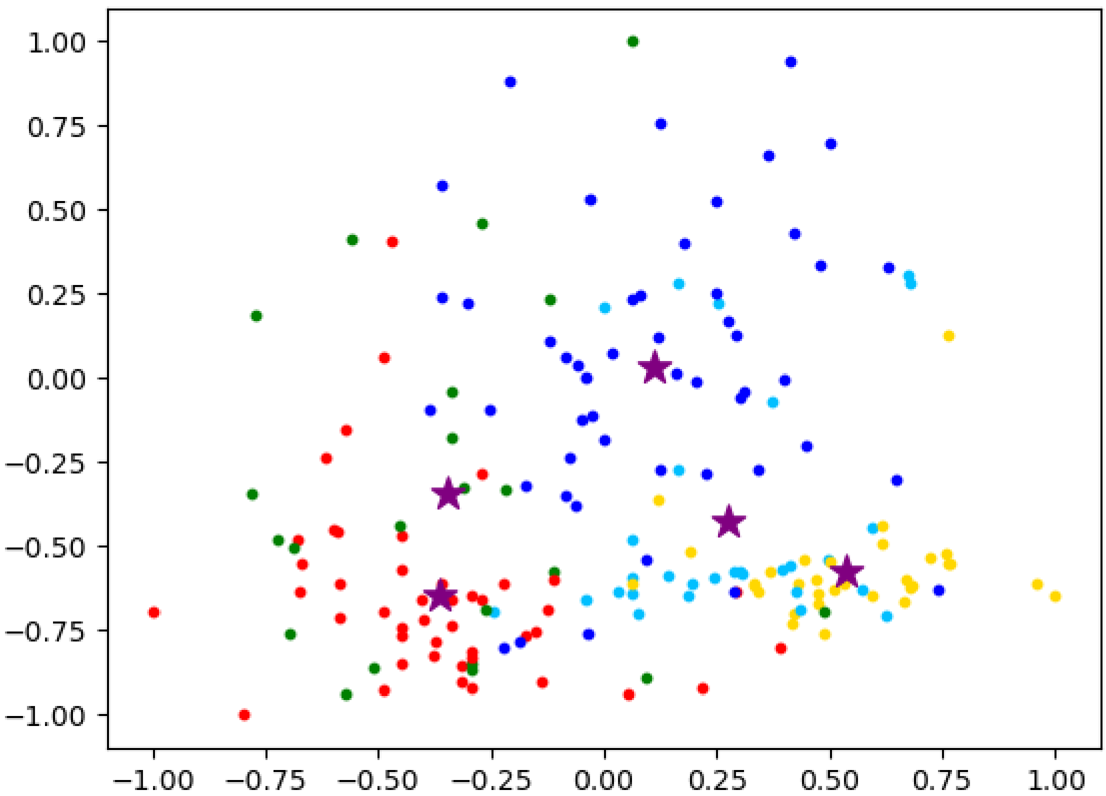
4个簇结果如图



Rand Index:0.8950041261981845

Silhouette Coefficient:0.25995492195881664

5个簇结果如图



Rand Index:0.8555195835713832
Silhouette Coefficient:0.18794414508559015

k	兰德系数RI	轮廓系数SI
2	0.6810131403542182	0.2987221815974774
3	0.9348695486573986	0.30089385185001355
4	0.8950041261981845	0.25995492195881664
5	0.8555195835713832	0.18794414508559015

可以看出，不进行PCA降维的话得到的图只能反应某几个维度的结果，看起来就非常零散，图像无法作为参考。而考察其兰德系数和轮廓系数，兰德系数要高于PCA处理后的结果，而轮廓系数要低于处理后的结果，这说明不经过PCA处理可以使得结果与真实结果的相似度更高因为其保留的信息更多，但同时分类轮廓也更不明显。