1. 数据集

采用全部数据进行聚类

```
# 导入数据集
dataset = pd.read_csv('iris.csv')
```

2. 手写K-means算法

kmeans算法手写源码

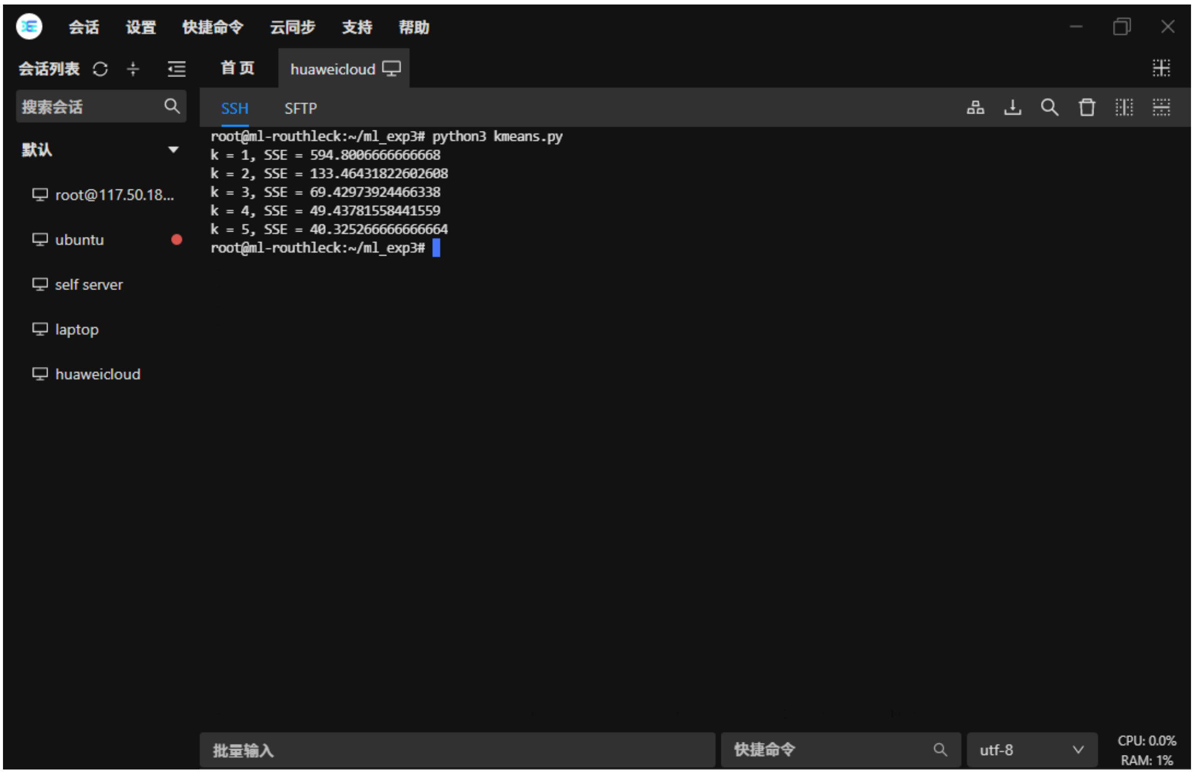
```
class KMeans:
    # 初始化
    def __init__(self, k=3, max_iters=300):
        self.k = k
        self.max_iters = max_iters
        # 质心
        self.centroids = None
        # 标签
        self.labels = None
        # 分类
        self.clusters = None
        # 损失值
        self.loss_value = []
        # SSE
        self.SSE = 0

    def _init_centroids(self, X):
        # 随机初始化质心
        return X[np.random.choice(X.shape[0], self.k, replace=False)]

    def _get_labels(self, X):
        # 计算距离
        distances = np.sqrt(np.sum((X - self.centroids[:, np.newaxis])**2,
axis=2))
        # 返回最近的质心索引
        return np.argmin(distances, axis=0)

    def _get_centroids(self, X):
        # 更新质心
        centroids = np.zeros((self.k, X.shape[1]))
        for i in range(self.k):
            centroids[i] = np.mean(X[self.labels == i], axis=0)
        return centroids

    def _get_clusters(self, X):
        # 更新SSE
        clusters = [[] for _ in range(self.k)]
        for i in range(self.k):
            clusters[i] = X[self.labels == i]
```

```python
        return clusters

    def plot(self):
        # 可视化
        ax = plt.axes(projection='3d')
        for i in range(self.k):
            ax.scatter3D(self.clusters[i][:, 0], self.clusters[i][:, 1], self.clusters[i][:, 2], label='cluster {}'.format(i))
        ax.scatter3D(self.centroids[:, 0], self.centroids[:, 1], self.centroids[:, 2], marker='*', c='black', label='centroids')
        ax.set_xlabel('sepal length')
        ax.set_ylabel('sepal width')
        ax.set_zlabel('petal length')

    # 训练
    def fit(self, X):
        # 初始化质心
        self.centroids = self._init_centroids(X)
        # 初始化标签
        self.labels = np.zeros((X.shape[0], 1))
        # 初始化分类
        self.clusters = [[] for _ in range(self.k)]
        # 迭代
        for _ in range(self.max_iters):
            # 更新标签
            self.labels = self._get_labels(X)
            # 更新质心
            new_centroids = self._get_centroids(X)
            if np.all(self.centroids == new_centroids):
                break
            self.centroids = self._get_centroids(X)
            # 更新SSE
            self.clusters = self._get_clusters(X)
            # 将损失值加入列表
            self.loss_value.append([_, self.loss(X=X)])

        # 计算SSE
        for i in range(self.k):
            self.SSE += np.sum((self.clusters[i] - self.centroids[i])**2)
        # 设置字体为黑体
        plt.rcParams['font.sans-serif'] = ['SimHei']
        # 设置标题为k的值
        self.plot()
        plt.title('k = {}时的聚类结果'.format(self.k))
        # 展示损失值的折线图
        plt.figure()
        plt.plot([i[0] for i in self.loss_value], [i[1] for i in self.loss_value])
        plt.title('k = {}时的损失值折线图'.format(self.k))

    # 预测
    def predict(self, X):
        return self._get_labels(X)

    # 损失值
```

```python
    def loss(self, X):
        loss = 0
        # 代价函数为畸变函数
        for i in range(X.shape[0]):
            # 找到最近的质心
            n_centroid = self.centroids[self.labels[i]]
            # 计算距离
            loss += np.sum((X[i] - n_centroid)**2)
        loss = loss / X.shape[0]
        return loss
```

华为云服务器运行结果

使用aechoterm ssh工具连接到华为云服务器

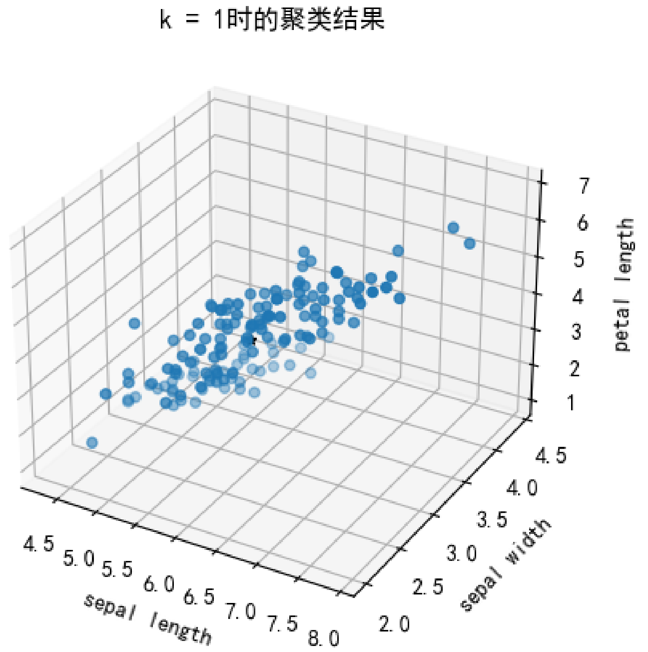


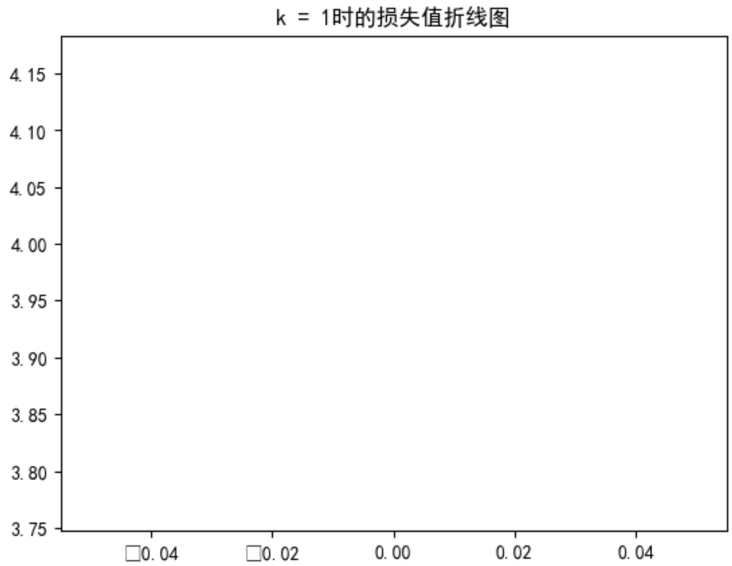
输出的图像文件:


远程 隐藏文件 刷新
/ root ml_exp3
名称 大小 类型 修改时间
..
iris.csv 4.86K 文件 2022-12-18 10:44:16
k=1_loss.png 14.51K 文件 2022-12-18 11:00:07
k=1_result.png 78.32K 文件 2022-12-18 11:00:06
k=2_loss.png 18.32K 文件 2022-12-18 11:00:07
k=2_result.png 89.21K 文件 2022-12-18 11:00:07
k=3_loss.png 25.21K 文件 2022-12-18 11:00:07
k=3_result.png 88.34K 文件 2022-12-18 11:00:07
k=4_loss.png 20.51K 文件 2022-12-18 11:00:08
k=4_result.png 94.65K 文件 2022-12-18 11:00:07
k=5_loss.png 21.67K 文件 2022-12-18 11:00:08
k=5_result.png 92.48K 文件 2022-12-18 11:00:08
kmeans.py 4.90K 文件 2022-12-18 10:48:38
SSE.png 18.26K 文件 2022-12-18 11:00:08

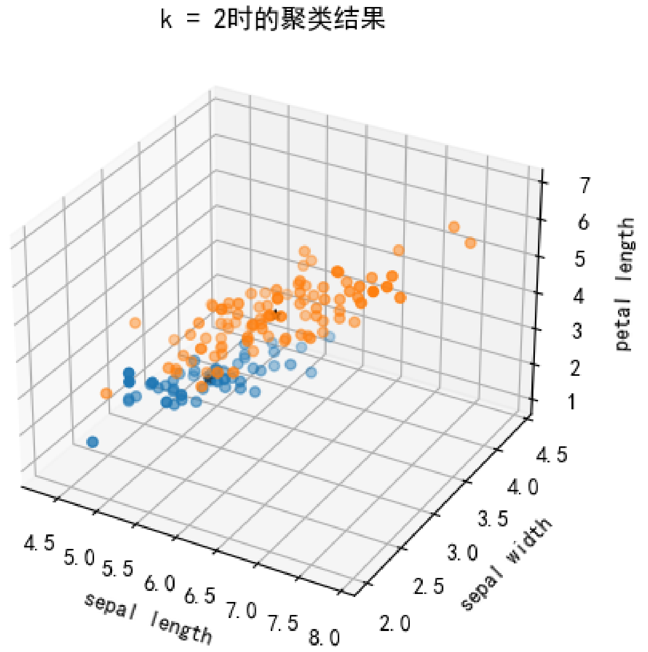
3. 实验结果可视化

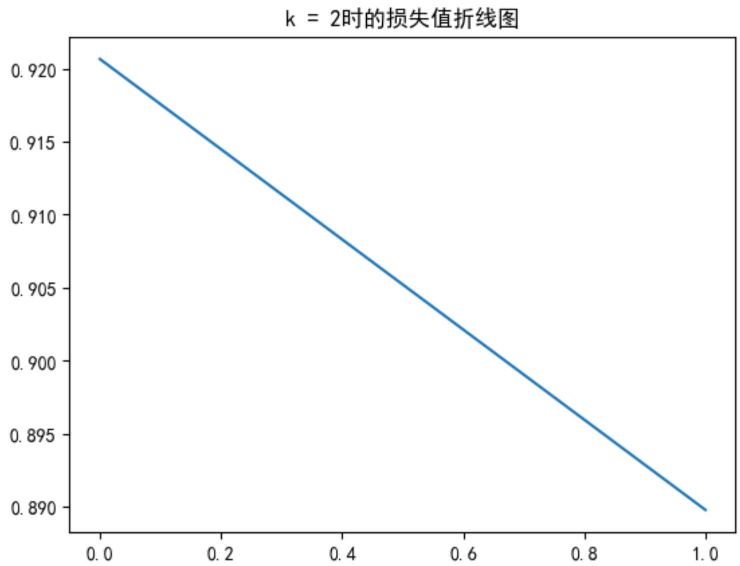
k=1


k = 1时的聚类结果
petal length
sepal width
sepal length
7
6
5
4
3
2
1
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

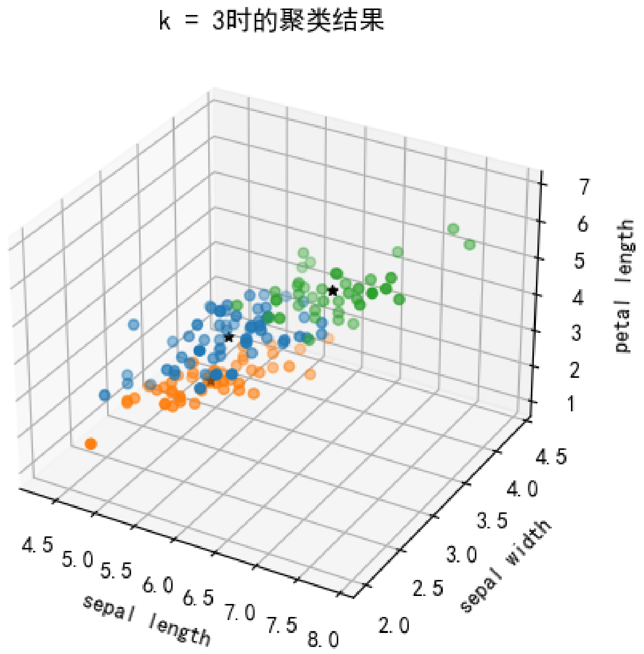

k = 1时的损失值折线图
4.15
4.10
4.05
4.00
3.95
3.90
3.85
3.80
3.75
□0.04
□0.02
0.00
0.02
0.04

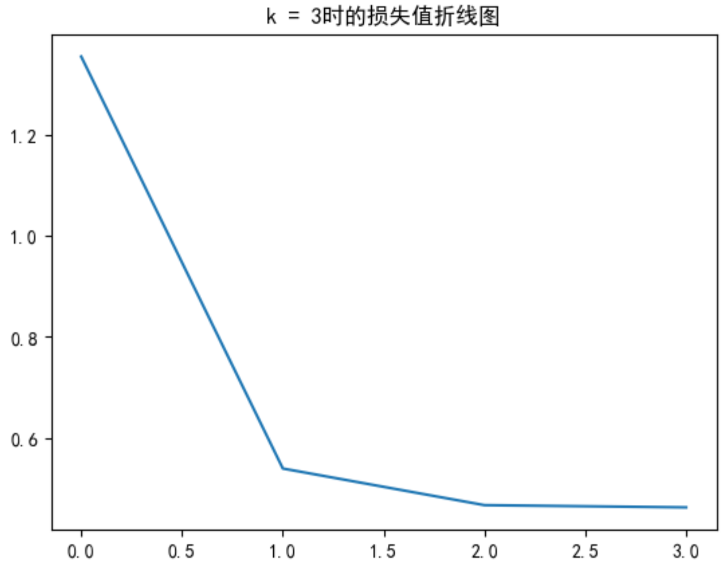
k=2


k = 2时的聚类结果
sepal length
sepal width
petal length
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
2.0
2.5
3.0
3.5
4.0
4.5
1
2
3
4
5
6
7


k = 2时的损失值折线图
0.920
0.915
0.910
0.905
0.900
0.895
0.890
0.0
0.2
0.4
0.6
0.8
1.0

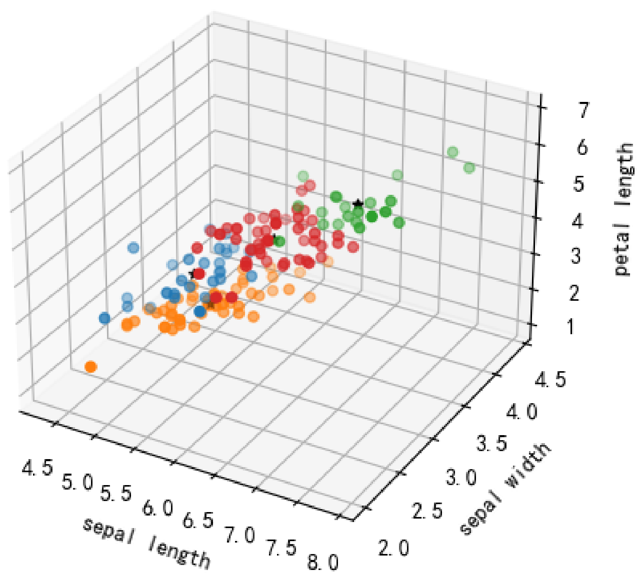
k=3


k = 3时的聚类结果
sepal length
sepal width
petal length
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
2.0
2.5
3.0
3.5
4.0
4.5
1
2
3
4
5
6
7

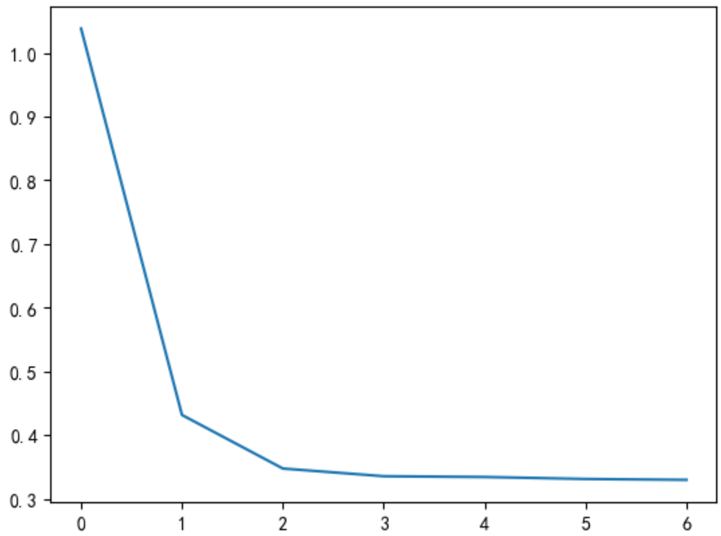

k = 3时的损失值折线图
1.2
1.0
0.8
0.6
0.0
0.5
1.0
1.5
2.0
2.5
3.0

k=4

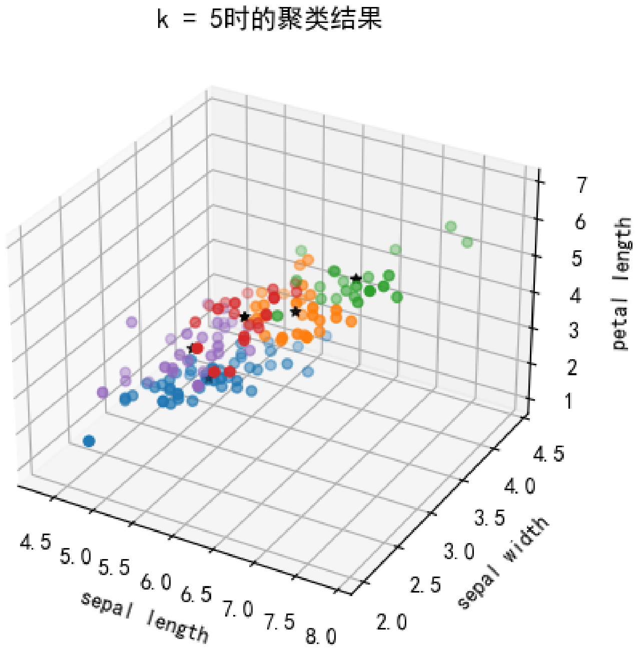
k = 4时的聚类结果

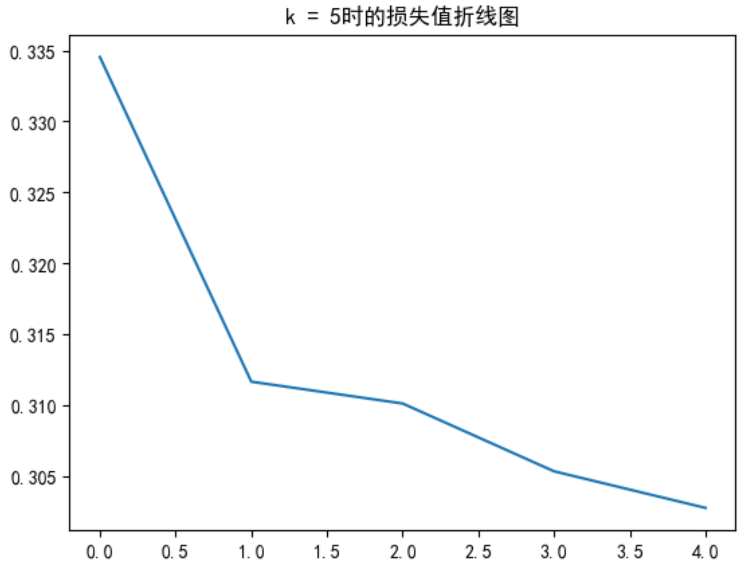


k = 4时的损失值折线图

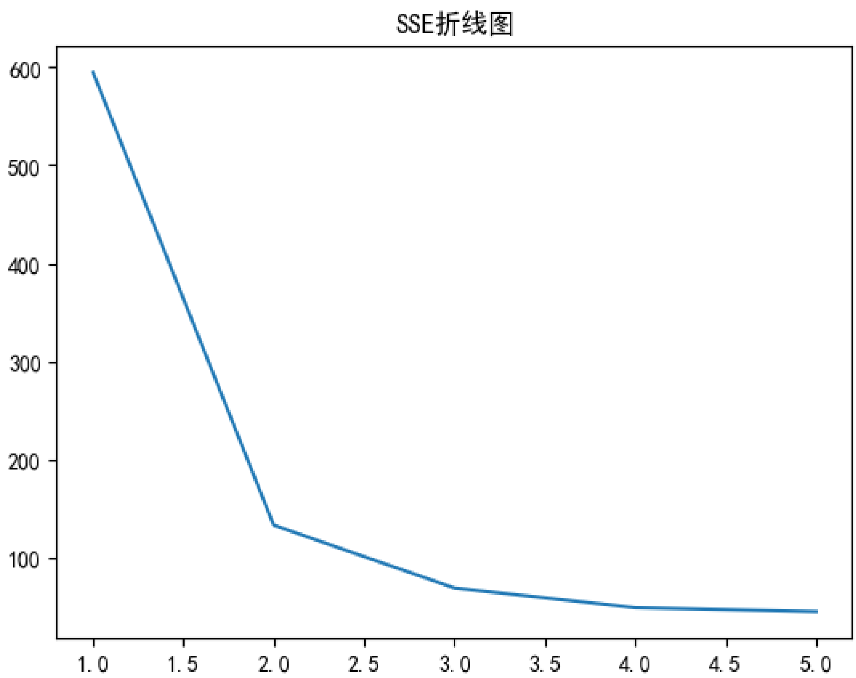


k=5


k = 5时的聚类结果
petal length
sepal width
sepal length


k = 5时的损失值折线图

4. 分析实验结果


SSE折线图

根据SSE折线图我们可以看到，当k=3后SSE的下降幅度明显趋向于缓慢，可以看做k=3为肘点，所以最佳k值为3