Homework3 实验报告

姓名：	程倩楠	学号：	201834858
实验课程：	Data Mining	实验名称：	Clustering with sklearn
指导老师：	尹建华	完成时间：	2018. 12. 08

一、实验目的

1. 理解各种聚类算法的原理。
2. 学会调用 sklearn 模块中的各种聚类函数和评估函数。

二、实验环境

python3.7

三、实验内容

1. 测试 sklearn 中以下聚类算法在 Tweets 数据集上的聚类效果。

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers

2. 使用 NMI(Normalized Mutual Information)作为评价指标，对各种聚类算法进行比较。

四、实验过程

（一）数据处理

定义一个数据处理函数，读取 json 文件，返回 sklearn 聚类函数及评估函数可直接调用的数据形式，包括以下几个步骤：

1. 读取 josn 数据文件，得到文本列表[‘text1’,’text2’,’text3’,...]和标签数组[label1 label2 label3 ...]。
2. 调用 sklearn 模块中的 TfidfVectorizer 函数从文本 list 中提取特征，获取 tweets 文本的表示向量，得到一个[n_samples,n_features]维度的稀疏矩阵，每一行对应一个文本表示向量。

TfidfVectorizer 函数实参

```
    #文本向量化
    #Extracting features from the dataset, using a sparse vectorizer
    #Vectorizer results are normalized
    vectorizer=TfidfVectorizer(max_df=0.5, #过滤掉df>(0.5*doc_num)的单词
                               max_features=3000, #构建词汇表仅考虑max_features
                               min_df=2, #过滤掉df<2的单词
                               stop_words='english', #过滤掉英文停用词
                               use_idf=True) #启动idf重新计算权重
    X=vectorizer.fit_transform(data) #稀疏矩阵
```

（二）聚类算法调用与评估

定义函数，分别调用 sklearn 中的八种聚类算法对 Tweets 数据进行聚类并返回聚类标签。利用 sklearn 自带的 NMI 评估函数对每个聚类算法聚类效果进行单独评价，得到评估分数。

1. K-Means

```
#定义函数实现KMeans聚类算法
# @param X 数据
# @param k 簇的数目
# @param minibatch 布尔型(True:MiniBatchKMeans False:KMeans)
# @return y_pred,km
def KMeansAlgorithm(X, k, minibatch):
    if minibatch:
        km=MiniBatchKMeans(n_clusters=k, #形成的簇数以及生成的中心数
                           init='k-means++', #用智能的方式选择初始聚类中心以加速收敛
                           n_init=3, #随机初始化的次数
                           batch_size=100, #Size of the mini batches
                           )
    else:
        km=KMeans(n_clusters=k,
                  init='k-means++',
                  max_iter=300, #一次单独运行的最大迭代次数
                  n_init=10, #使用不同的聚类中心进行初始化的次数
                  )
    km.fit(X)
    y_pred=km.labels_
    return y_pred,km
```

2. Affinity Propagation

```python
#定义函数实现AffinityPropagation聚类算法
# @param X 样本特征数据
# @return y_pred
def AffinityPropagationAlgorithm(X):
    '''
    1. 参数damping: float, optional, default: 0.5
       Damping factor (between 0.5 and 1) is the extent to which the current
       value is maintained relative to incoming values (weighted 1 - damping).
       This in order to avoid numerical oscillations when updating these values
       (messages).
    2. 参数preference: array-like, shape (n_samples,) or float, optional
       The number of exemplars, ie of clusters, is influenced by the input
       preferences value.
       If the preferences are not passed as arguments, they will be set to the
       median of the input similarities.
    '''
    ap=AffinityPropagation(damping=0.5, preference=None)
    ap.fit(X)
    y_pred=ap.labels_
    return y_pred
```

3. Mean-shift

```python
#定义函数执行AffinityPropagation聚类算法
# @param X 样本特征数据
# @return y_pred
def MeanShiftAlgorithm(X):
    '''
    参数bandwidth: float, optional
    Bandwidth used in the RBF kernel.
    '''
    ms=MeanShift(bandwidth=0.9)
    ms.fit(X)
    y_pred=ms.labels_
    return y_pred
```

4. Spectral Clustering

```python
#定义函数执行SpectralClustering聚类算法
# @param X 样本特征数据
# @param k 簇的数目
# @return y_pred
def SpectralClusteringAlgorithm(X,k):
    #参数n_clusters: integer, optional
    #The dimension of the projection subspace.
    sc=SpectralClustering(n_clusters=k)
    sc.fit(X)
    y_pred=sc.labels_
    return y_pred
```

5. Ward Hierarchical Clustering

```python
#定义函数执行WardHierarchicalClustering聚类算法
# @param X 样本特征数据
# @param k 簇的数目
```

```
# @return y_pred
def WardHierarchicalClusteringAlgorithm(X,k):
    #参数linkage: optional (default="ward")
    #ward minimizes the variance of the clusters being merged.
    #参数n_clusters: int, default=2
    #The number of clusters to find.
    whc=AgglomerativeClustering(linkage='ward',
                                n_clusters=k)
    whc.fit(X)
    y_pred=whc.labels_
    return y_pred
```

6. Agglomerative Clustering

```
#定义函数执行AgglomerativeClustering聚类算法
# @param X 样本特征数据
# @param k 簇的数目
# @return y_pred
def AgglomerativeClusteringAlgorithm(X,k):
    '''
    1. 参数linkage:  {"ward", "complete", "average", "single"}, optional
       average uses the average of the distances of each observation of the
       two sets.
    2. 参数n_clusters: int, default=2
       The number of clusters to find.
    '''
    ac=AgglomerativeClustering(linkage='average',
                               n_clusters=k)
    ac.fit(X)
    y_pred=ac.labels_
    return y_pred
```

7. DBSCAN

```
#定义函数执行DBSCAN聚类算法
# @param X 样本特征数据
# @return y_pred
def DBSCANAlgorithm(X):
    '''
    1. 参数eps: float, optional
       The maximum distance between two samples for them to be considered as in
       the same neighborhood.
    2. 参数min_samples: int, optional
       The number of samples (or total weight) in a neighborhood for a point to
       be considered as a core point.
       This includes the point itself.
    '''
    ds=DBSCAN(eps=0.5,
              min_samples=1,
              metric='cosine')
    ds.fit(X)
    y_pred=ds.labels_
    return y_pred
```

8. Gaussian Mixture

```python
#定义函数执行GaussianMixture聚类算法
# @param X 样本特征数据
# @param k 簇的数目
# @return y_pred
def GaussianMixtureAlgorithm(X,k):
    #参数n_components: int, defaults to 1
    #The number of mixture components.
    gm=GaussianMixture(n_components=k)
    gm.fit(X)
    y_pred=gm.predict(X)
    return y_pred
```

（三）比较 8 种聚类算法

主文件 main.py 循环调用 8 种聚类算法，得到在 Tweets 数据集上的聚类标签，对 NMI 评估分数进行直观的比较。主要包括以下过程：

1. 调用数据处理函数，得到 sklearn 函数可以直接调用的数据格式。
2. 定义聚类算法列表，循环调用 8 种聚类算法对 Tweets 文本进行聚类，得到聚类标签，将 NMI 评估分数保存到评估列表。
3. 绘制条形图，直观比较 8 种聚类算法的效果。
4. 排序并输出聚类算法及其对应的评估分数。

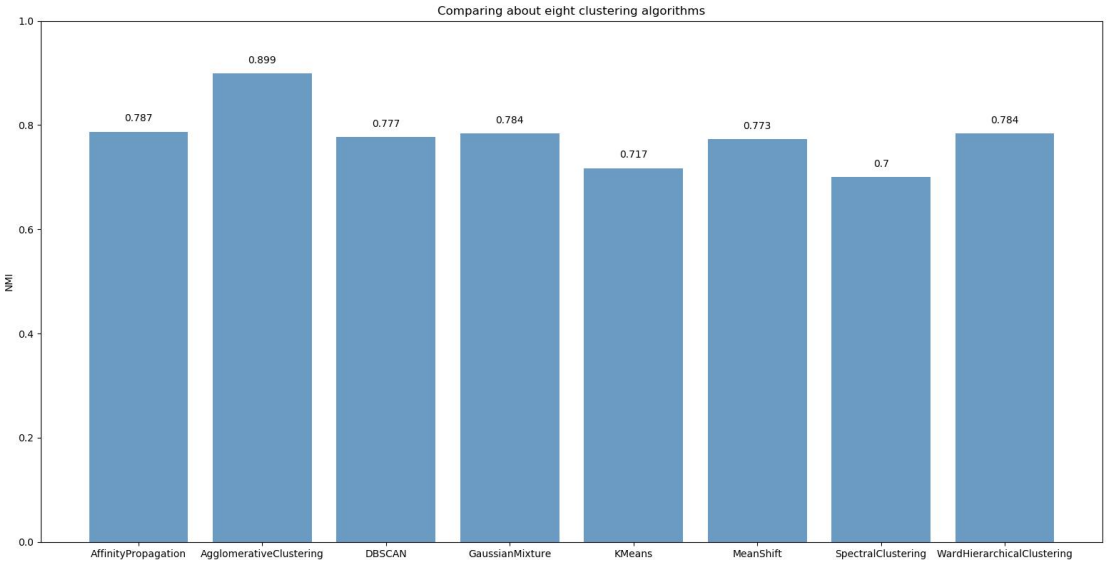
五、实验结果

（一）单独调用和评估聚类算法

聚类算法	NMI
K-Means	0.708
Affinity Propagation	0.787
Mean-shift	0.773
Spectral Clustering	0.708
Ward Hierarchical Clustering	0.784
Agglomerative Clustering	0.899
DBSCAN	0.777
Gaussian Mixture	0.789

（二）循环调用和比较聚类算法

1. 绘制条形图，直观比较 8 种聚类算法的效果。其中，横坐标代表 8 种聚类算法，纵坐标代表 NMI 评估分数。



2. 依据 NMI 评估分数对算法进行排序

```
Clustering algorithm and corresponding NMI score:
1. AgglomerativeClustering: 0.899
2. AffinityPropagation: 0.787
3. GaussianMixture: 0.784
4. WardHierarchicalClustering: 0.784
5. DBSCAN: 0.777
6. MeanShift: 0.773
7. KMeans: 0.717
8. SpectralClustering: 0.7
```