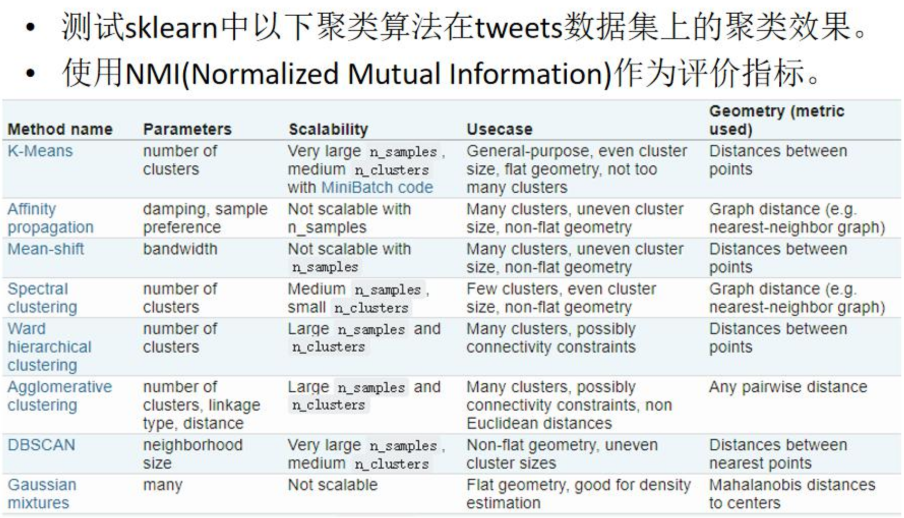
Homework 5 实验报告

实验内容：

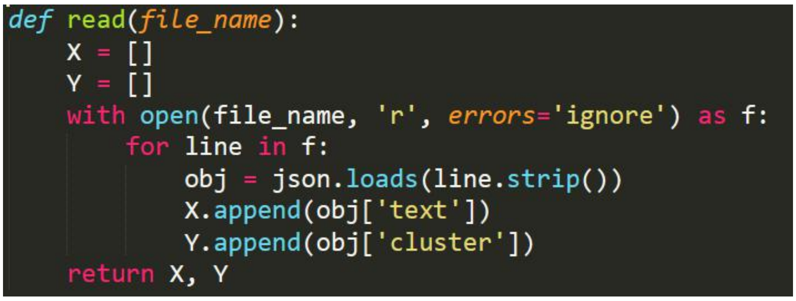


- 测试sklearn中以下聚类算法在tweets数据集上的聚类效果。
- 使用NMI(Normalized Mutual Information)作为评价指标。

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

实验步骤：

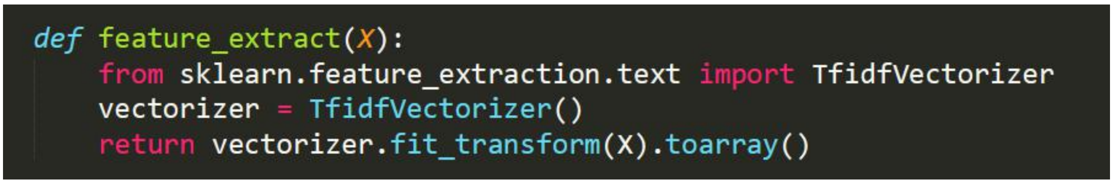
1. 读入 tweets



```python
def read(file_name):
    X = []
    Y = []
    with open(file_name, 'r', errors='ignore') as f:
        for line in f:
            obj = json.loads(line.strip())
            X.append(obj['text'])
            Y.append(obj['cluster'])
    return X, Y
```

2. 将文本表示为向量

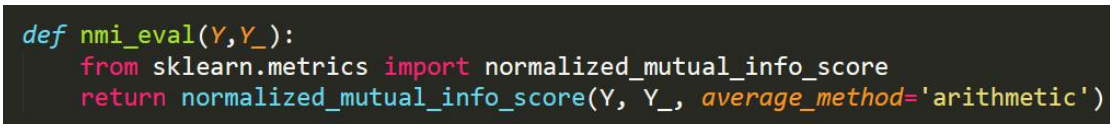
用 sklearn 里 feature_extraction 模块里的 TfidfVectorizer 将文本转换为 tfidf 的向量



```python
def feature_extract(X):
    from sklearn.feature_extraction.text import TfidfVectorizer
    vectorizer = TfidfVectorizer()
    return vectorizer.fit_transform(X).toarray()
```

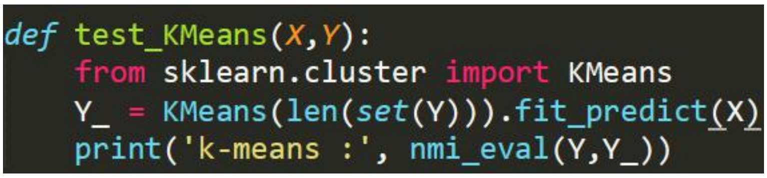
3. 测试 sklearn 的各种聚类效果

用 nmi 评估：



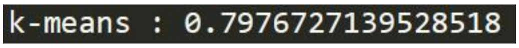
```
def nmi_eval(Y,Y_):
    from sklearn.metrics import normalized_mutual_info_score
    return normalized_mutual_info_score(Y, Y_, average_method='arithmetic')
```

测试 k-means:



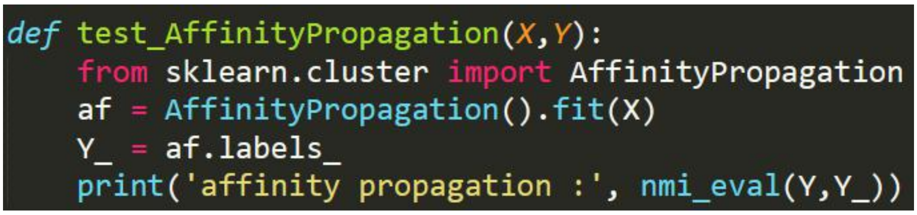
```
def test_KMeans(X,Y):
    from sklearn.cluster import KMeans
    Y_ = KMeans(len(set(Y))).fit_predict(X)
    print('k-means :', nmi_eval(Y,Y_))
```

结果:



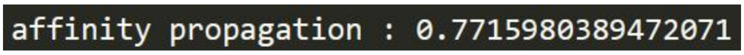
```
k-means : 0.7976727139528518
```

测试 affinity propagation：



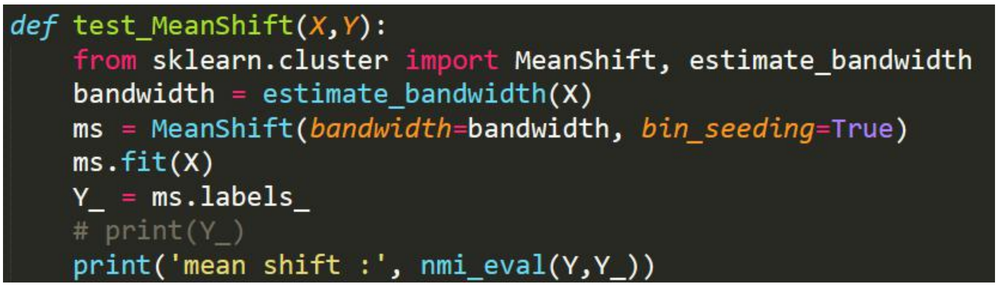
```
def test_AffinityPropagation(X,Y):
    from sklearn.cluster import AffinityPropagation
    af = AffinityPropagation().fit(X)
    Y_ = af.labels_
    print('affinity propagation :', nmi_eval(Y,Y_))
```

结果：



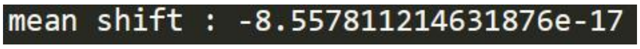
```
affinity propagation : 0.7715980389472071
```

测试 mean shift：



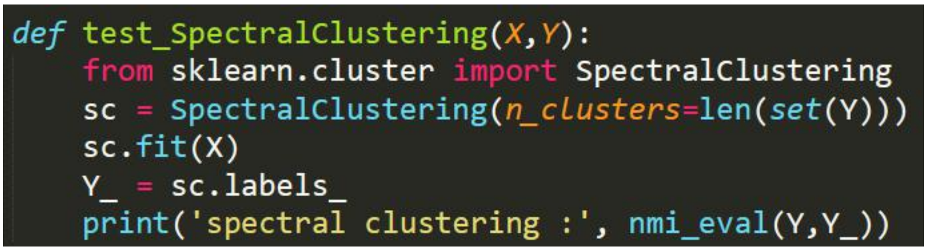
```
def test_MeanShift(X,Y):
    from sklearn.cluster import MeanShift, estimate_bandwidth
    bandwidth = estimate_bandwidth(X)
    ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)
    ms.fit(X)
    Y_ = ms.labels_
    # print(Y_)
    print('mean shift :', nmi_eval(Y,Y_))
```

结果：



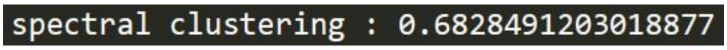
```
mean shift : -8.557811214631876e-17
```

测试 spectral clustering：



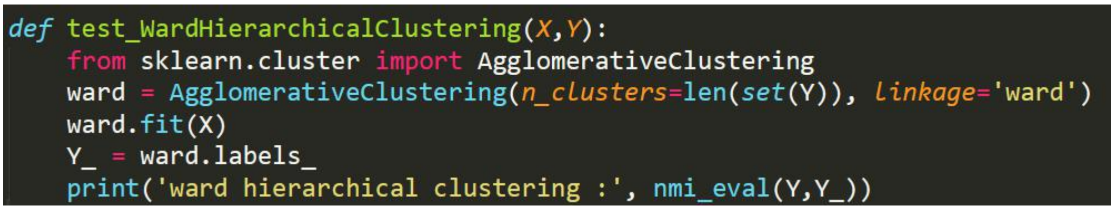
```
def test_SpectralClustering(X,Y):
    from sklearn.cluster import SpectralClustering
    sc = SpectralClustering(n_clusters=len(set(Y)))
    sc.fit(X)
    Y_ = sc.labels_
    print('spectral clustering :', nmi_eval(Y,Y_))
```

结果：



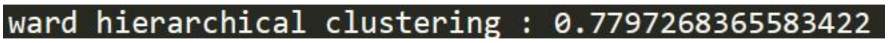
```
spectral clustering : 0.6828491203018877
```

测试 ward hierarchical clustering：



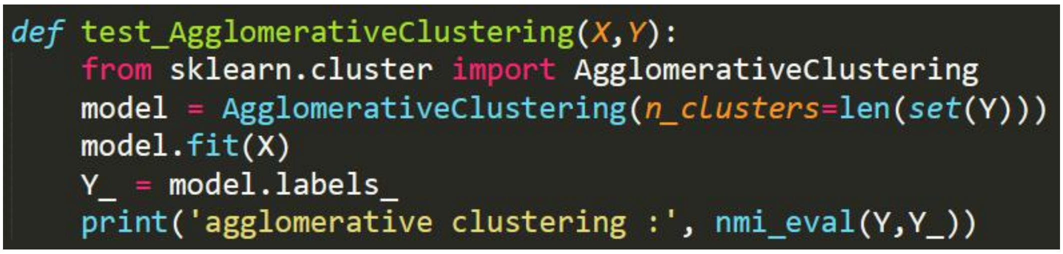
```python
def test_WardHierarchicalClustering(X,Y):
    from sklearn.cluster import AgglomerativeClustering
    ward = AgglomerativeClustering(n_clusters=len(set(Y)), linkage='ward')
    ward.fit(X)
    Y_ = ward.labels_
    print('ward hierarchical clustering :', nmi_eval(Y,Y_))
```

结果：



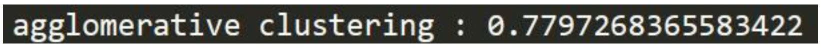
```
ward hierarchical clustering : 0.7797268365583422
```

测试 agglomerative clustering：



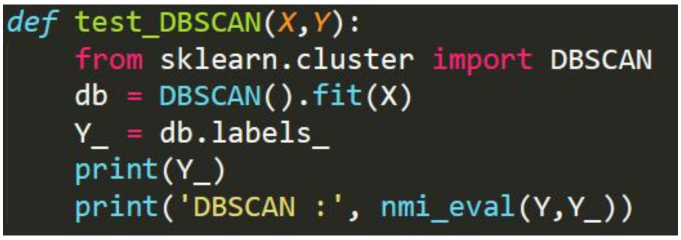
```python
def test_AgglomerativeClustering(X,Y):
    from sklearn.cluster import AgglomerativeClustering
    model = AgglomerativeClustering(n_clusters=len(set(Y)))
    model.fit(X)
    Y_ = model.labels_
    print('agglomerative clustering :', nmi_eval(Y,Y_))
```

结果：



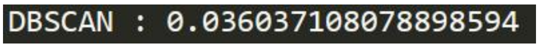
```
agglomerative clustering : 0.7797268365583422
```

测试 DBSCAN:



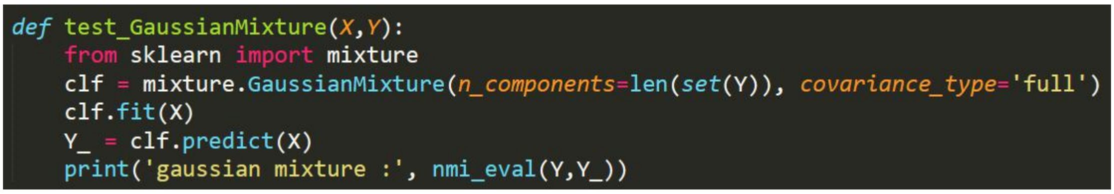
```python
def test_DBSCAN(X,Y):
    from sklearn.cluster import DBSCAN
    db = DBSCAN().fit(X)
    Y_ = db.labels_
    print(Y_)
    print('DBSCAN :', nmi_eval(Y,Y_))
```

结果：



```
DBSCAN : 0.036037108078898594
```

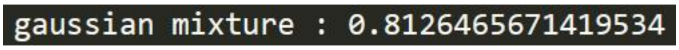
测试高斯混合：



```python
def test_GaussianMixture(X,Y):
    from sklearn import mixture
    clf = mixture.GaussianMixture(n_components=len(set(Y)), covariance_type='full')
    clf.fit(X)
    Y_ = clf.predict(X)
    print('gaussian mixture :', nmi_eval(Y,Y_))
```

运行报错，提示 MemoryError

只跑了前 500 个数据，结果：



```
gaussian mixture : 0.8126465671419534
```