Tsiounis Konstantinos, 2630
Polyzos Alexios, 2338

Patterns Recognition

2nd Exercises Set

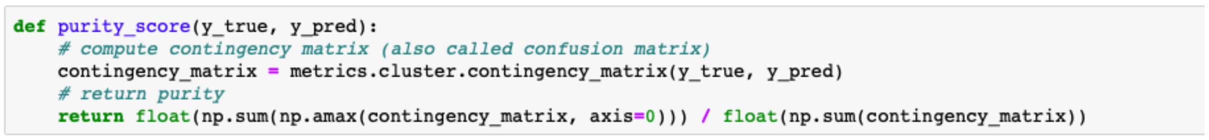
Introduction

We used Jupyter Notebook to build this project. If you don't already have jupyter notebook or anaconda on your computer, we suggest you to install Anaconda which includes everything you need to run our notebooks. For this project, we used Scikit-Learn Library for the models, Pandas Library and NumPy.

About our code, you have it in a zip but you can also see it on clicking here. If you can download it or run for some reason, you can open the html files in our GitHub repository to see these notebooks with their results.

Purity Score

After some research we did about purity score, we found that purity can be calculated using confusion matrix. So, to calculate purity, we first created a confusion matrix using sklearn's metric for clusters and we calculated the sum of the max values in each row divided by the sum of all the values. To do this, we created a function as you can see below.



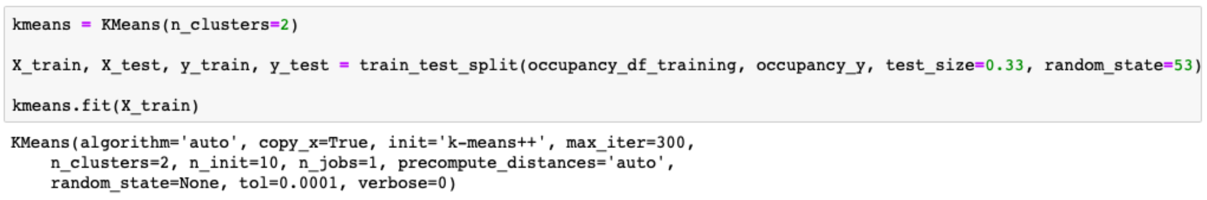
```
def purity_score(y_true, y_pred):
    # compute contingency matrix (also called confusion matrix)
    contingency_matrix = metrics.cluster.contingency_matrix(y_true, y_pred)
    # return purity
    return float(np.sum(np.amax(contingency_matrix, axis=0))) / float(np.sum(contingency_matrix))
```

F1 Score

The f1 score is calculated using sklearn's function from metrics package.

K-Means Clustering

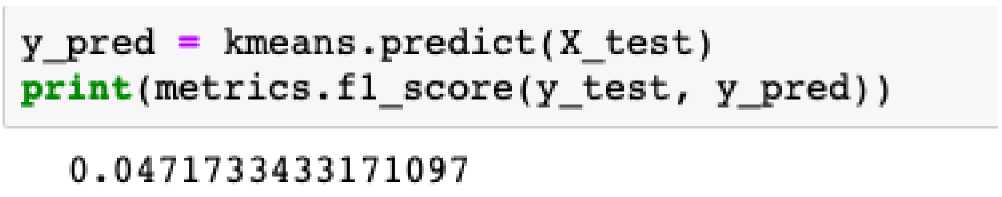
The data loading and pre-processing are the same with the previous exercises set so we won’t repeat this section. After these steps, we created a k-means classifier from SciKitLearn library using 2 clusters, we split our data into training and test data and we fit the model with the training data.



```
kmeans = KMeans(n_clusters=2)

X_train, X_test, y_train, y_test = train_test_split(occupancy_df_training, occupancy_y, test_size=0.33, random_state=53)

kmeans.fit(X_train)
```

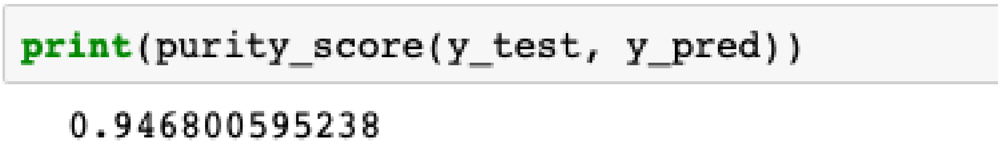
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

The last step is to compute purity and f1 score.



```
y_pred = kmeans.predict(X_test)
print(metrics.f1_score(y_test, y_pred))
```

```
0.0471733433171097
```



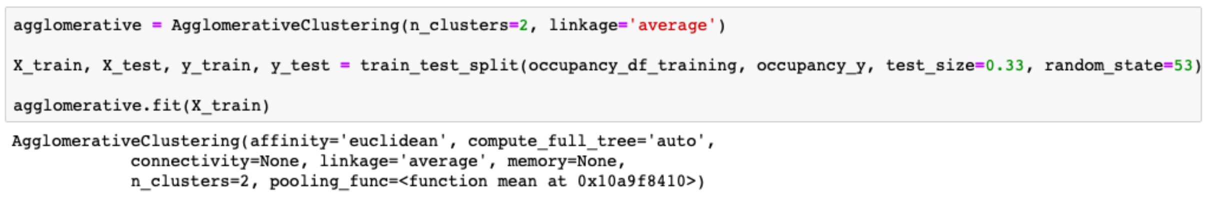
```
print(purity_score(y_test, y_pred))
```

```
0.946800595238
```

The above steps repeated for the spambase dataset.

Agglomerative Hierarchical Clustering

For the Agglomerative Hierarchical Clustering, we created an AgglomerativeClustering object with 2 clusters and linkage parameter equal to average. The other steps were the same with the other models.

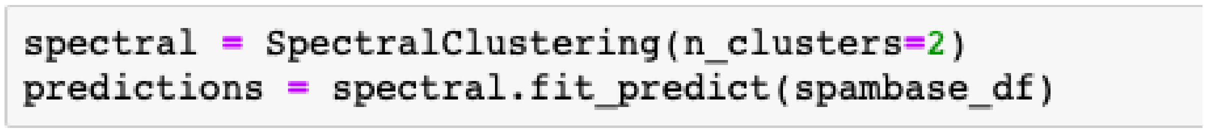


```
agglomerative = AgglomerativeClustering(n_clusters=2, linkage='average')

X_train, X_test, y_train, y_test = train_test_split(occupancy_df_training, occupancy_y, test_size=0.33, random_state=53)

agglomerative.fit(X_train)
```

```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
            connectivity=None, linkage='average', memory=None,
            n_clusters=2, pooling_func=<function mean at 0x10a9f8410>)
```

Spectral Clustering

For the Spectral Clustering, we created a SpectralClustering object with 2 clusters. The other steps were the same with the other models.



```
spectral = SpectralClustering(n_clusters=2)
predictions = spectral.fit_predict(spambase_df)
```

However, because of the datasets size we couldn’t complete the clustering. As we found after research, Spectral Clustering is difficult to run with big arrays of data. We also tried to run on cloud but we couldn’t reach to an end. So, below at the results, the results for spectral will be empty.

Results

	2 Clusters		4 Clusters		8 Clusters	
	Purity	F1	Purity	F1	Purity	F1
Occupancy Detection	0.946	0.047	0.936	0.023	0.971	0.038
Spam Dataset	0.647	0.512	0.667	0.595	0.706	0.544

Table 1. K-Means Results

	2 Clusters		4 Clusters		8 Clusters	
	Purity	F1	Purity	F1	Purity	F1
Occupancy Detection	0.805	0.201	0.978	0.207	0.978	0.034
Spam Dataset	0.617	0.553	0.633	0.366	0.633	0.618

Table 2. Agglomerative Hierarchical Clustering Results

	2 Clusters		4 Clusters		8 Clusters	
	Purity	F1	Purity	F1	Purity	F1
Occupancy Detection	-	-	-	-	-	-
Spam Dataset	-	-	-	-	-	-

Table 3. Spectral Clustering Results

As we can see at results above, K-means performs better and between the two datasets it performs better with Occupancy Detection dataset. Moreover, we can see that increasing the number of clusters, the results are getting way better. However, there is a very low F1 score and by increasing the clusters the score is going down which is expected cause in some way cause F1 score is better in binary classification. On the other hand, purity doesn't work well for imbalanced data: if a size 1000 dataset consists of two classes, one class contains 999 points and the other has only 1 point. No matter how bad a clustering algorithm performs, it will always give a very high purity value. So, between K-Means Clustering & Agglomerative Hierarchical Clustering, the winner is K-Means on these datasets.