Travail Pratique

Classer des chiffres manuscrits en exploitant l'algorithme des K-moyennes

YOU Borachhun, LE Do Thanh Dat

Importer des libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, confusion_matrix, \
                            classification_report
from scipy.cluster.hierarchy import dendrogram, linkage, cut_tree
```

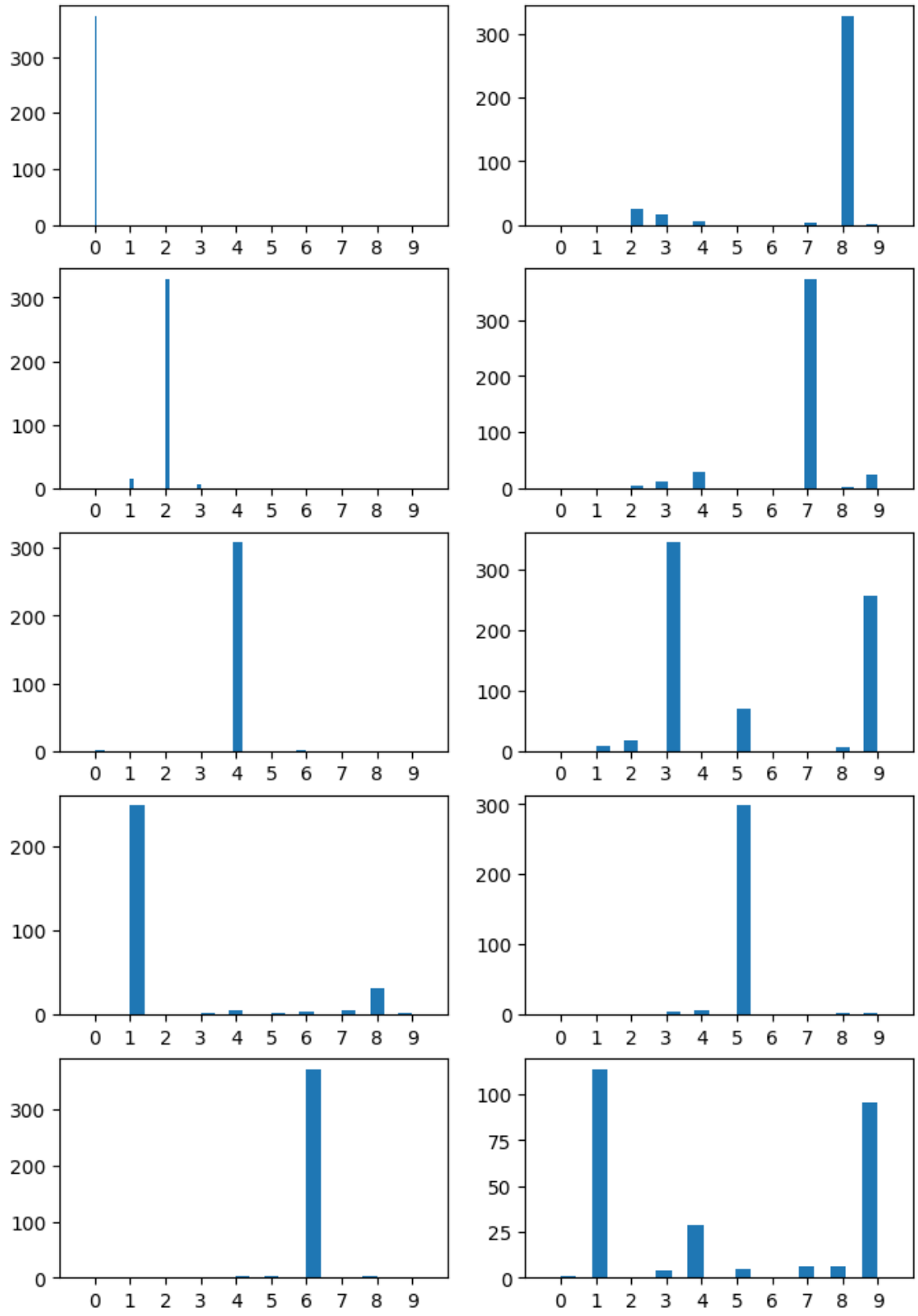
I. Apprentissage

1. Faire un K-moyennes avec K=10 sur la base d'apprentissage (BA) : optdigits.tra

In [2]:
```
train_data = pd.read_csv('optdigits.tra', header=None)
X_train = train_data.iloc[:, 0:64]
y_train = train_data.iloc[:, 64]

kmeans = KMeans(n_clusters=10, n_init='auto', random_state=31)\
            .fit(X_train)
cluster_train = kmeans.labels_
```

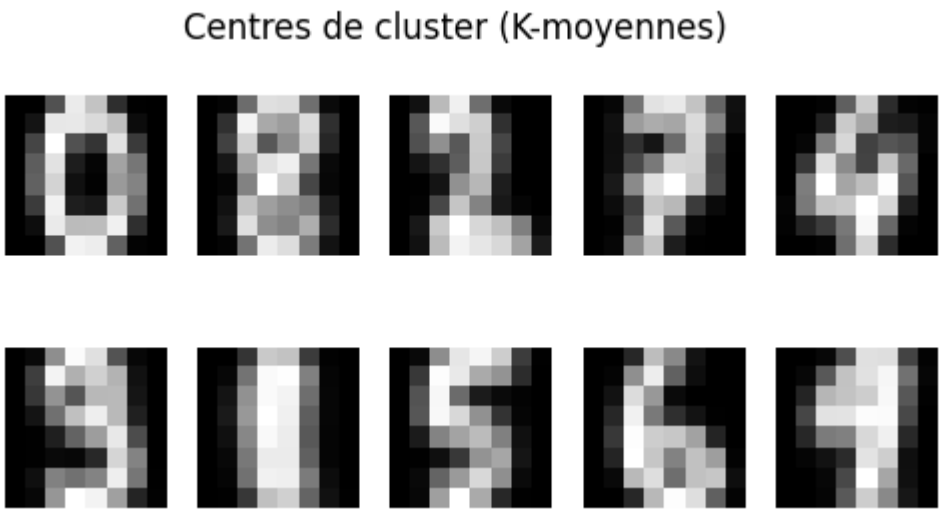
2. Par cluster : faire un histogramme du nombre de chiffres de chaque classe

In [3]:
```
plt.figure(figsize=(8,12))
for i in range(10):
    plt.subplot(5, 2, i+1)
    plt.xlim(-1,10)
    plt.xticks(range(0,10))
    plt.hist(y_train[[c == i for c in cluster_train]], bins=20)
plt.show()
```



In [4]:

```
# Plot center of each cluster
kmeans_centers = kmeans.cluster_centers_
plt.figure(figsize=(6,3))
plt.suptitle('Centres de cluster (K-moyennes)')
for i in range(len(kmeans_centers)):
    plt.subplot(2, 5, i+1)
    plt.imshow(np.reshape(kmeans_centers[i], (8,8)), cmap=plt.get_cmap('gray'))
    plt.axis('off')
plt.show()
```



Tous clusters sont bien classés, sauf 2 clusters :

- Le 6ème cluster : le chiffre 3 se trouve le plus, mais il y a aussi beaucoup de chiffre 9.
- Le 10ème cluster : le chiffre 1 se trouve le plus, mais il y a aussi beaucoup de chiffre 9.

Par conséquent, le centre des deux clusters semble être un mélange de deux chiffres (3 et 9, 1 et 9, respectivement).

Autres remarques :

- ~2/3 des chiffres 1 sont dans le 7ème cluster et l'autre ~1/3 sont dans le 10ème cluster.
- ~2/3 des chiffres 9 sont dans le 6ème cluster et l'autre ~1/3 sont dans le 10ème cluster.

3. Mesurer la qualité du Clustering avec l'indice de la Silhouette

In [5]:
```
print("Silhouette index:", silhouette_score(X_train, cluster_train))
```

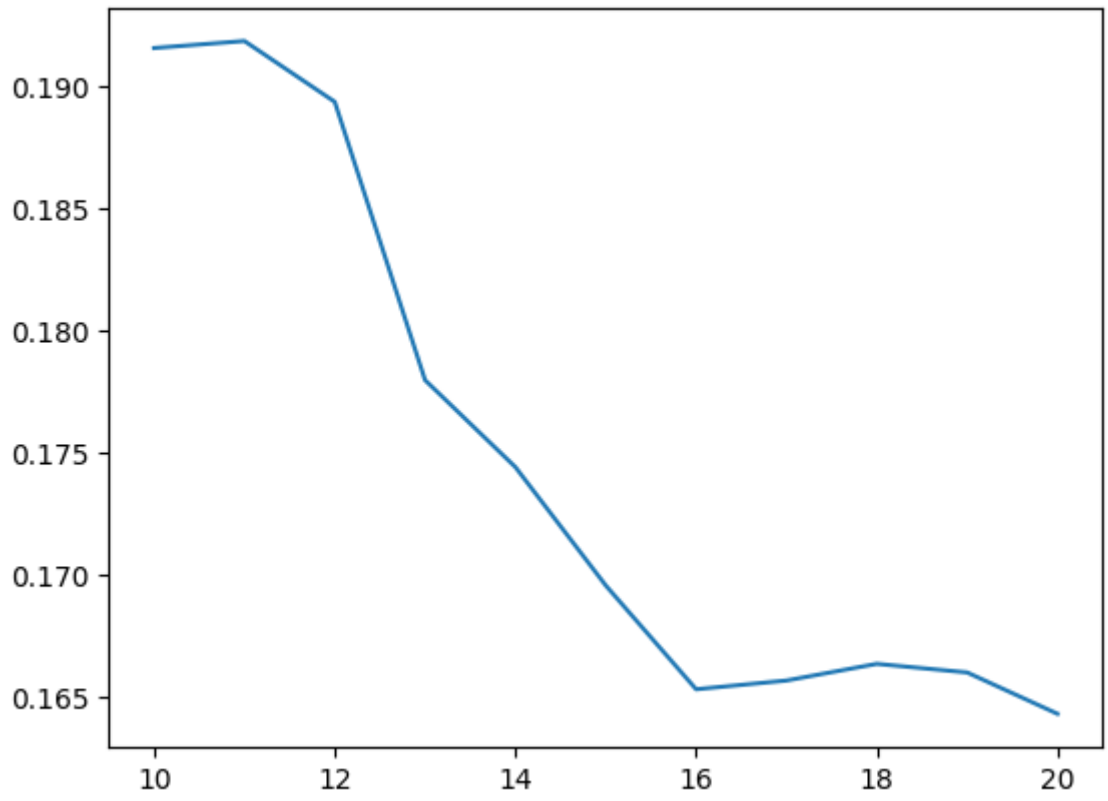
```
Silhouette index: 0.19155548990533083
```

Globalement, ce n'est pas un très bon clustering puisque l'indice de la Silhouette moyen est loin de 1. De plus, l'indice est proche de 0, ce qui signifie qu'il y a des clusters qui se chevauchent (overlapping clusters), ce que nous pouvons voir dans la question précédente.

4. Faire varier K entre 10 et 20 clusters et calculer pour chaque K l'indice de la Silhouette

In [6]:
```
sil_scores = [
    silhouette_score(
        X_train,
        KMeans(n_clusters=K, n_init='auto', random_state=31) \
            .fit(X_train).labels_
    )
    for K in range(10,20+1)
```

```
]
plt.plot([K for K in range(10,20+1)], sil_scores)
plt.show()

print("Maximum value of Silhouette index:", max(sil_scores))
```




```
Maximum value of Silhouette index: 0.19184242102758556
```

Selon le graphique ci-dessus, nous obtenons le meilleur clustering avec K = 11.

II. Test

1. Par cluster : faire un vote à la majorité pour attribuer un label à chaque cluster

In [7]:
```
cluster_to_label = {}

# for each cluster
for i in range(10):

    # labels in the cluster
    label_in_cluster_i = y_train[[c == i for c in cluster_train]]

    # count number of each label in the cluster
    num_label_in_cluster_i = [
        np.sum([int(l == j) for l in label_in_cluster_i])
        for j in range(10)
    ]

    # assign label that exist the most in the cluster to the cluster
    cluster_to_label[i] = num_label_in_cluster_i.index(
                                max(num_label_in_cluster_i))
```

```
print('Majority vote (cluster to label):\n', cluster_to_label)
```

```
Majority vote (cluster to label):
 {0: 0, 1: 8, 2: 2, 3: 7, 4: 4, 5: 3, 6: 1, 7: 5, 8: 6, 9: 1}
```

Remarque : par le vote à la majorité, nous pouvons voir qu'il n'y a pas de cluster attribué au label 9 pour ce clustering, et il y a deux clusters attribués au label 1.

2. Pour chaque élément de la BT (Base de Test) : optdigits.tes
   - Chercher le Cluster (Centre) le plus proche
   - Attribuer à cet élément de la BT le label associé au Cluster le plus proche
   - Calculer la matrice de confusions (matrice 10x10) et la performance globale : analyser les confusions

In [8]:
```
test_data = pd.read_csv('optdigits.tes', header=None)
X_test = test_data.iloc[:, 0:64]
y_test = test_data.iloc[:, 64]

cluster_test = kmeans.predict(X_test)
y_test_predict = [cluster_to_label[c] for c in cluster_test]

# Confusion matrix
print('Confusion matrix:\n{}'.format(
                    confusion_matrix(y_test, y_test_predict)))

# Performance
print(classification_report(y_test, y_test_predict, digits=3))
```

```
Confusion matrix:
[[176   0   0   0   2   0   0   0   0   0]
 [  0 156  21   1   0   1   3   0   0   0]
 [  1   5 149   9   0   0   0   4   9   0]
 [  0   1   0 163   0   2   0  10   7   0]
 [  0   9   0   0 160   0   0   7   5   0]
 [  0   1   0  26   1 153   1   0   0   0]
 [  1   4   0   0   0   0 175   0   1   0]
 [  0   8   0   0   1   1   0 166   3   0]
 [  0  28   1  10   0   2   1   1 131   0]
 [  0  24   0 145   0   4   0   4   3   0]]
              precision    recall  f1-score   support

           0      0.989     0.989     0.989       178
           1      0.661     0.857     0.746       182
           2      0.871     0.842     0.856       177
           3      0.460     0.891     0.607       183
           4      0.976     0.884     0.928       181
           5      0.939     0.841     0.887       182
           6      0.972     0.967     0.970       181
           7      0.865     0.927     0.895       179
           8      0.824     0.753     0.787       174
           9      0.000     0.000     0.000       180

    accuracy                          0.795      1797
   macro avg      0.756     0.795     0.766      1797
weighted avg      0.755     0.795     0.766      1797
```

```
C:\Users\Admin\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Admin\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Admin\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

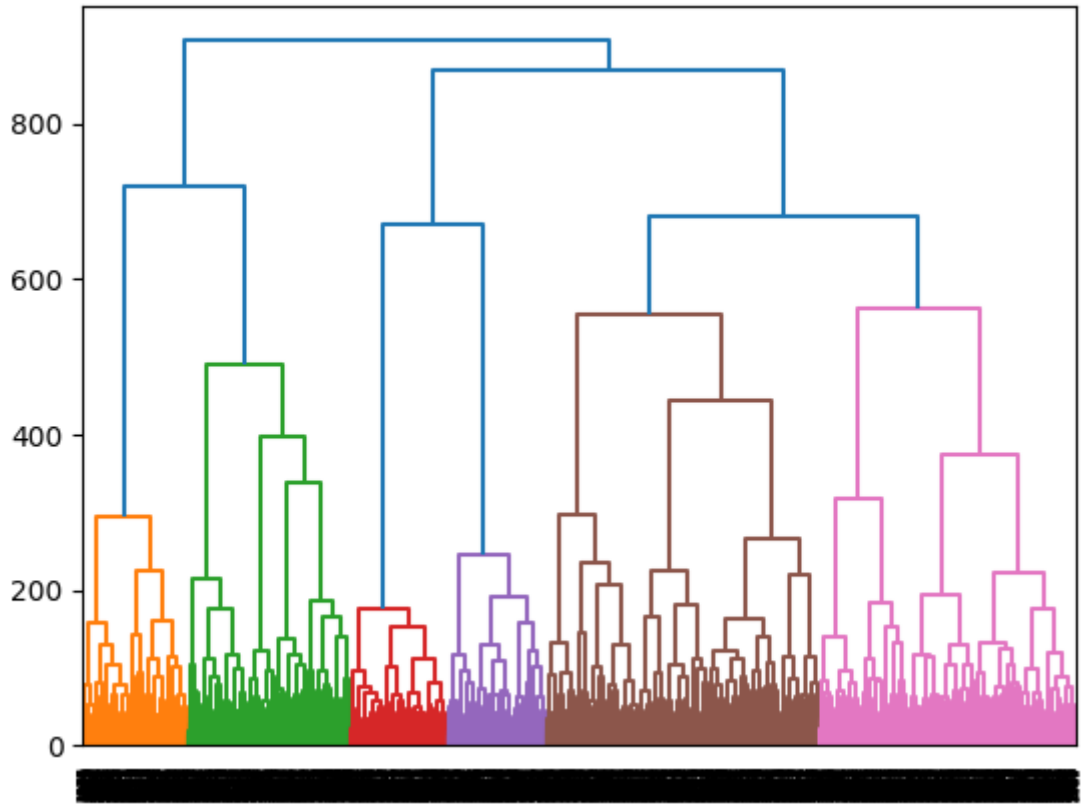
Selon la matrice de confusion, nous pouvons voir que le résultat est bon sauf pour le label 9. Pour les labels 0 à 8, les éléments diagonaux de la matrice ont de grandes valeurs, ce qui signifie que beaucoup de ces chiffres sont correctement classés. Cependant, tous les chiffres 9 sont mal classés comme autres chiffres, avec la majorité étant 3. Ceci est la conséquence des questions précédentes de n'avoir aucun cluster attribué au label 9 et beaucoup de chiffre 9 étant dans le cluster du chiffre 3.

III. Comparaison au Clustering Hiérarchique (avec le critère de Ward)

1. Phase d’apprentissage : sur optdigits.tra

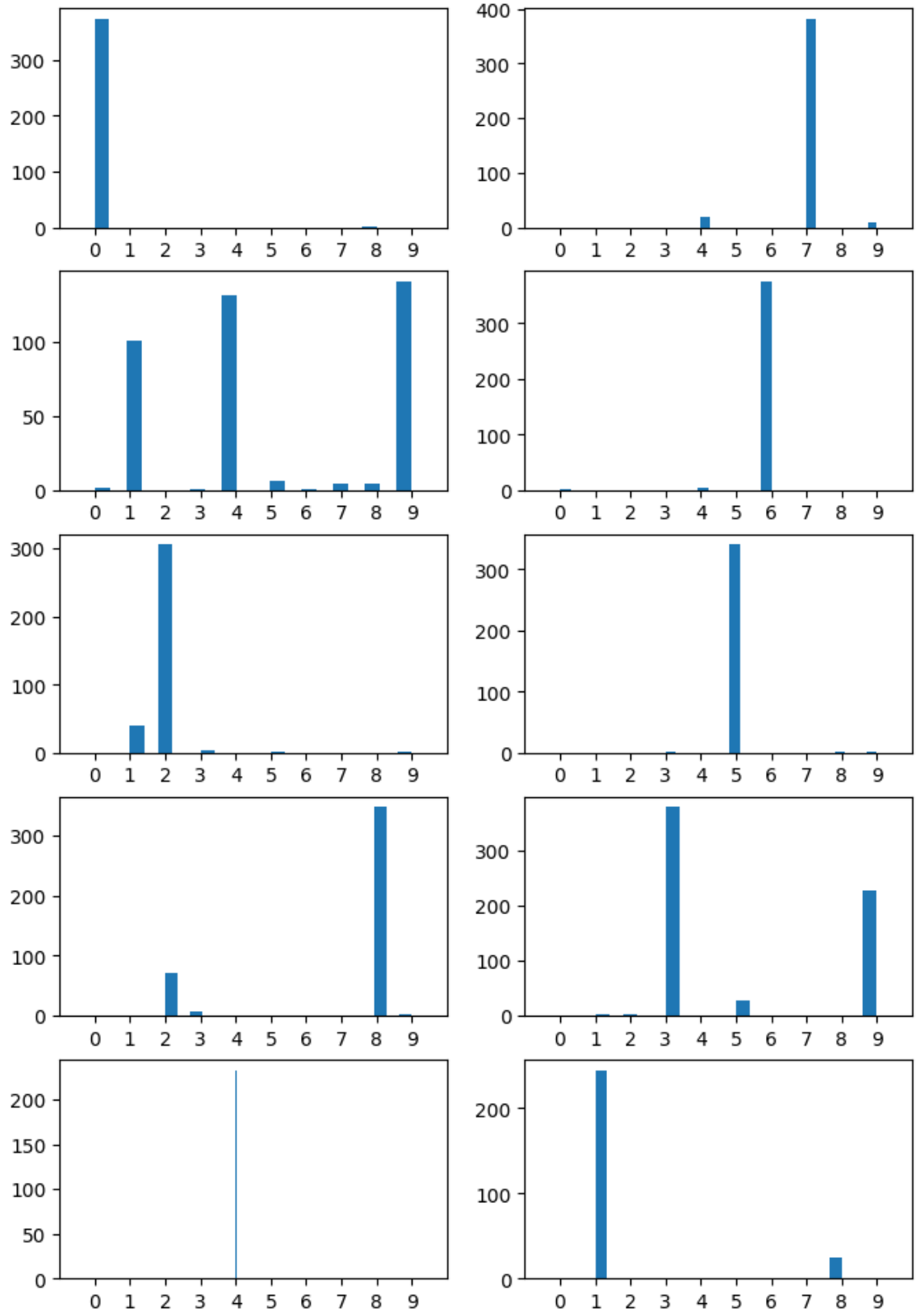
- Faire un Clustering Hiérarchique et visualiser le dendrogramme

In [9]:
```
linkage_matrix = linkage(X_train, method='ward', metric='euclidean')
d = dendrogram(linkage_matrix)
plt.show()
```



- Couper le dendrogramme à K=10, calculer l'indice de la Silhouette et faire les histogrammes par cluster (à comparer avec histogrammes avec K-moyennes). Comparer à la valeur de la Silhouette obtenue avec l'algorithme des K-moyennes.

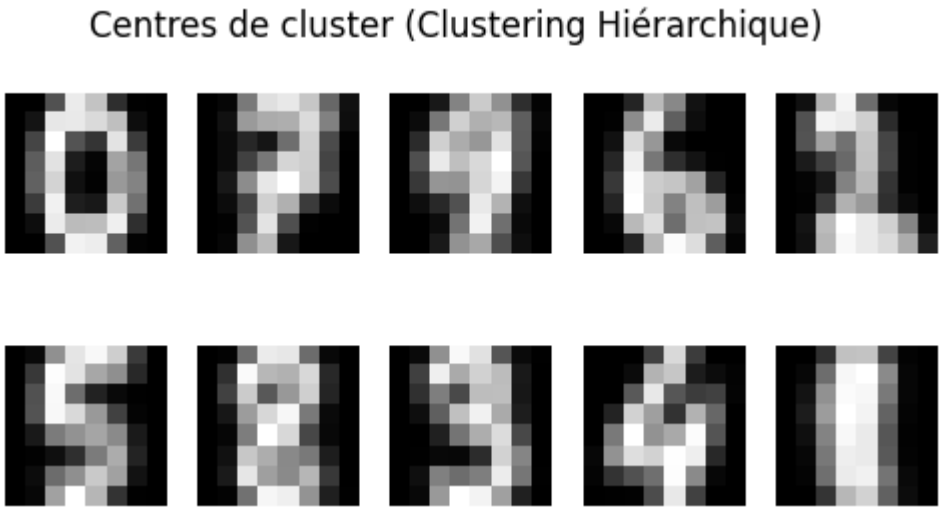
In [10]:
```
# Cut the dendrogram with K = 10 clusters
hierarchy_labels = cut_tree(linkage_matrix, n_clusters=10).ravel()

# Faire des histogrammes
plt.figure(figsize=(8,12))
for i in range(10):
    plt.subplot(5, 2, i+1)
    plt.xlim(-1,10)
    plt.xticks(range(0,10))
    plt.hist(y_train[[c == i for c in hierarchy_labels]], bins=20)
plt.show()
```



In [11]:

```python
# Trouver les centres de chaque cluster
centers = []
for i in range(10):
    sum = np.zeros(64)
    count = 0
    for j in range(len(hierarchy_labels)):
        if hierarchy_labels[j] == i:
            sum += X_train.loc[j,:]
            count += 1
    centers.append(sum/count)
```

```
# Plot center of each cluster
plt.figure(figsize=(6,3))
plt.suptitle('Centres de cluster (Clustering Hiérarchique)')
for i in range(len(centers)):
    plt.subplot(2, 5, i+1)
    plt.imshow(np.reshape(np.array(centers[i]), (8,8)), cmap=plt.get_cmap('gray'
    plt.axis('off')
plt.show()
```



Comme la méthode de K-moyennes, il y a 2 mauvais clusters :

- Le 3ème cluster : le chiffre 9 se trouve le plus, mais il y a aussi beaucoup de chiffre 1 et 4.
- Le 8ème cluster : le chiffre 3 se trouve le plus, mais il y a aussi beaucoup de chiffre 9.

Par conséquent, le centre des deux clusters semble être un mélange de deux ou trois chiffres.

Autres remarques :

- ~1/3 des chiffres 1 sont dans le 3ème cluster et l'autre ~2/3 sont dans le 10ème cluster.
- ~1/3 des chiffres 4 sont dans le 3ème cluster et l'autre ~2/3 sont dans le 9ème cluster.
- ~1/3 des chiffres 9 sont dans le 3ème cluster et l'autre ~2/3 sont dans le 8ème cluster.

In [12]:
```
# Mesurer la qualité du Clustering avec l'indice de la Silhouette
print("Silhouette index:", silhouette_score(X_train, hierarchy_labels))
```

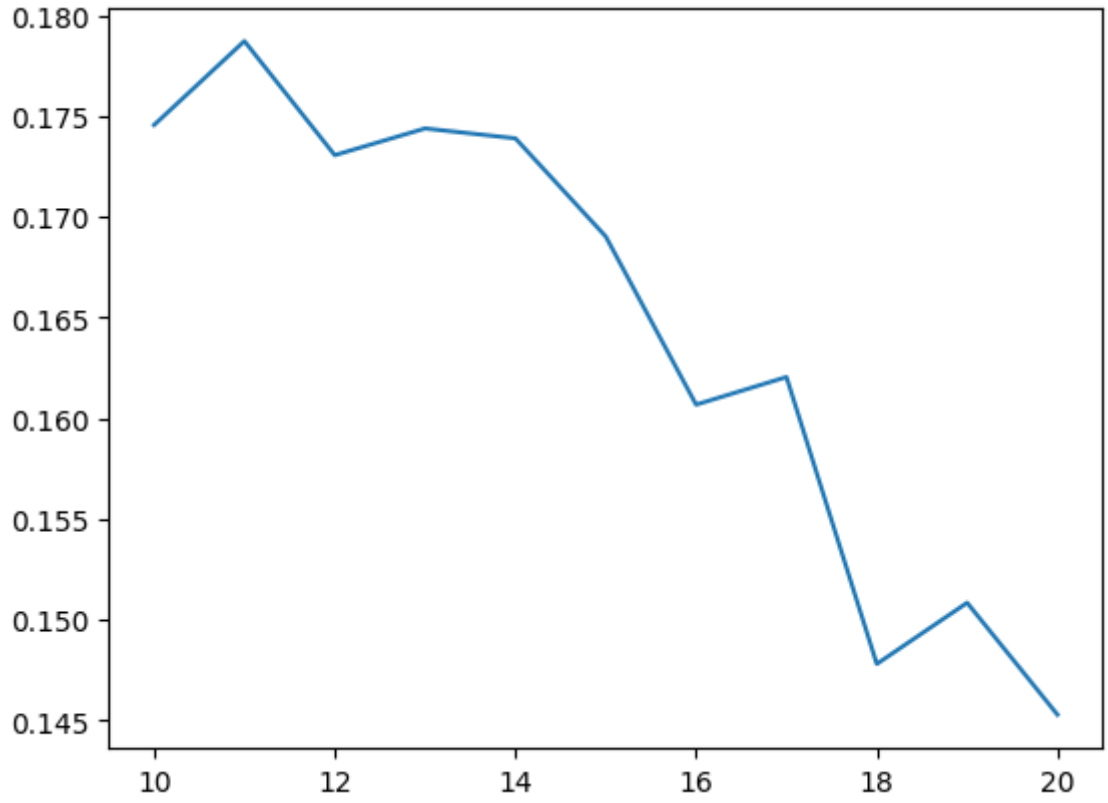
```
Silhouette index: 0.1745470931891432
```

Pour l'indice de la Silhouette moyen, il est presque égal à celui du K-moyennes, ce qui signifie que le clustering n'est pas très bon.

- Couper le dendrogramme à d'autres niveaux hiérarchiques: entre 11 et 20 clusters et calculer pour chaque K l'indice de la Silhouette. Pour quelle valeur de K obtenez-

vous la meilleure partition ? Comparer au K-moyennes.

In [14]:
```
# Couper le dendrogramme à d'autres niveaux hiérarchiques: entre 11 et 20
# clusters et calculer pour chaque K l'indice de la Silhouette
sil_scores = [
    silhouette_score(
        X_train,
        cut_tree(linkage_matrix, n_clusters=K).ravel()
    )
    for K in range(10,20+1)
]
plt.plot([K for K in range(10,20+1)], sil_scores)
plt.show()

print("Maximum value of Silhouette index:", max(sil_scores))
```




```
Maximum value of Silhouette index: 0.17871212582446985
```

Selon le graphique ci-dessus, K = 11 est le meilleur K pour le Clustering Hiérarchique car il a l'indice de la Silhouette le plus élevé, et c'est la même valeur de K du K-moyennes.

2. Phase de test après Clustering Hiérarchique : sur optdigits.tes

(i) Par cluster : faire un vote à la majorité pour attribuer un label à chaque cluster (la classe la plus représentée dans chaque cluster)

In [15]:
```
cluster_to_label = {}

# for each cluster
for i in range(10):

    # labels in the cluster
    label_in_cluster_i = y_train[[c == i for c in hierarchy_labels]]
```

```
    # count number of each class in the cluster
    num_label_in_cluster_i = [
        np.sum([int(l == j) for l in label_in_cluster_i])
        for j in range(10)
    ]

    # assign label that exist the most in the cluster to the cluster
    cluster_to_label[i] = num_label_in_cluster_i.index(
                                max(num_label_in_cluster_i))

print('Majority vote (cluster to label):\n', cluster_to_label)
```

```
Majority vote (cluster to label):
 {0: 0, 1: 7, 2: 9, 3: 6, 4: 2, 5: 5, 6: 8, 7: 3, 8: 4, 9: 1}
```

Remarque : tous les labels sont attribués à un cluster, même le label 9, malgré les 2 mauvais clusters comme le K-moyennes.

(ii) Pour chaque élément de la BT (Base de Test) : optdigits.tes

- Chercher le Cluster (Centre) le plus proche
- Attribuer à cet élément de la BT le label associé au Cluster le plus proche
- Calculer la matrice de confusions (matrice 10x10) et la performance globale : analyser les confusions. Comparer les résultats de classification à ceux obtenus avec le K-moyennes. Analysez.

In [16]:
```
# Find the distances between each element and the centers
y_pred = []
for i in range(len(X_test)):
    dist = [
        math.dist(X_test.loc[i,:], centers[j])
        for j in range(10)
    ]
    # Attribuer à cet élément de la BT le label associé au Cluster le plus proch
    y_pred.append(dist.index(min(dist)))

# Convert cluster to label
y_pred = [cluster_to_label[c] for c in y_pred]

# Confusion matrix
print('Confusion matrix:\n{}'.format(
                    confusion_matrix(y_test, y_pred)))

# Calcul la performance globale
print(classification_report(y_test, y_pred, digits=3))
```

```
Confusion matrix:
[[176   0   0   0   2   0   0   0   0   0]
 [  0 105  24   0   0   1   2   0   0  50]
 [  1   3 141   7   0   0   0   2  22   1]
 [  0   1   0 162   0   2   0   9   9   0]
 [  0   4   0   0 127   0   3   3   5  39]
 [  0   0   0  13   1 166   1   0   0   1]
 [  1   3   0   0   1   1 175   0   0   0]
 [  0   0   0   0   0   0   0 156   2  21]
 [  0  18   1   4   0   1   1   1 137  11]
 [  0   0   0 143   0   2   0   3   4  28]]
              precision    recall  f1-score   support

           0      0.989     0.989     0.989       178
           1      0.784     0.577     0.665       182
           2      0.849     0.797     0.822       177
           3      0.492     0.885     0.633       183
           4      0.969     0.702     0.814       181
           5      0.960     0.912     0.935       182
           6      0.962     0.967     0.964       181
           7      0.897     0.872     0.884       179
           8      0.765     0.787     0.776       174
           9      0.185     0.156     0.169       180

    accuracy                          0.764      1797
   macro avg      0.785     0.764     0.765      1797
weighted avg      0.785     0.764     0.765      1797
```

Selon la matrice de confusion, comme le K-moyennes, nous pouvons voir que le résultat est bon sauf pour le label 9. Pour les labels 0 à 8, les éléments diagonaux de la matrice ont de grandes valeurs, ce qui signifie que beaucoup de ces chiffres sont correctement classés. Cependant, ce qui est différent du K-moyennes, c'est qu'il y a quelques chiffres 9 qui sont correctement classés même si la majorité des chiffres sont mal classés comme 3. Ceci est la conséquence de la question précédente d'avoir un cluster attribué au label 9.