Week 11 Quiz

Clarissa Tai - rt2822

Due Tues Nov 22nd, 11:59pm

The MNIST digits dataset is a dataset commonly used to demonstrate image recognition.

The dataset is composed of a set of images of handwritten digits from 0 to 9. There are 1797 images, each 8x8 pixels. If we flatten each image we get a dataset of 1797 observations, each with 64 features, each belonging to one of 10 classes.

In this quiz we'll see how well these images cluster using KMeans clustering. We'll compare the KMeans clustering assingments to clusters generated by HAC with Ward linkage.

Instructions

Replace the Name and UNI in cell above and the notebook filename

Replace all '__' below using the instructions provided.

When completed,

1. make sure you've replaced Name and UNI in the first cell and filename
2. Kernel -> Restart & Run All to run all cells in order
3. Print Preview -> Print (Landscape Layout) -> Save to pdf
4. post pdf to GradeScope

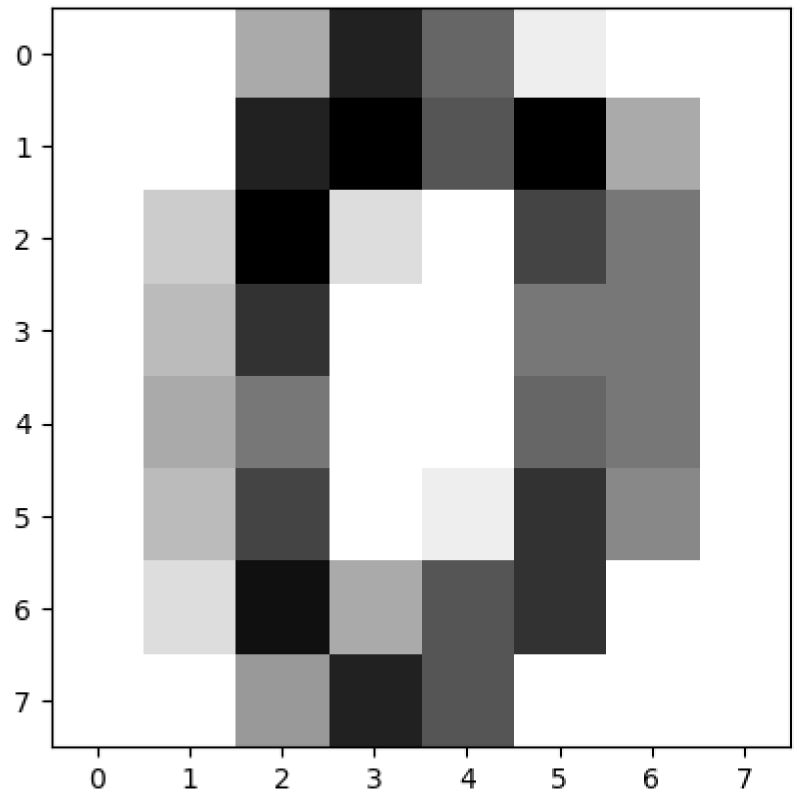
Setup Environment

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Load the Dataset

In [2]:
```
# From sklearn datasets import load_digits.
from sklearn.datasets import load_digits

# Load the dataset into 'digits' using load_digits
digits = load_digits()

# Extract digits['data'] to X_digits. No need to reshape as each image has already been flattened to 1x64
X_digits = digits['data']

# Extract the labels in digits['target'] to y_digits
y_digits = digits['target']

# Assert that the shape of X_digits is 1797 rows, 64 columns
assert X_digits.shape == (1797,64)
```

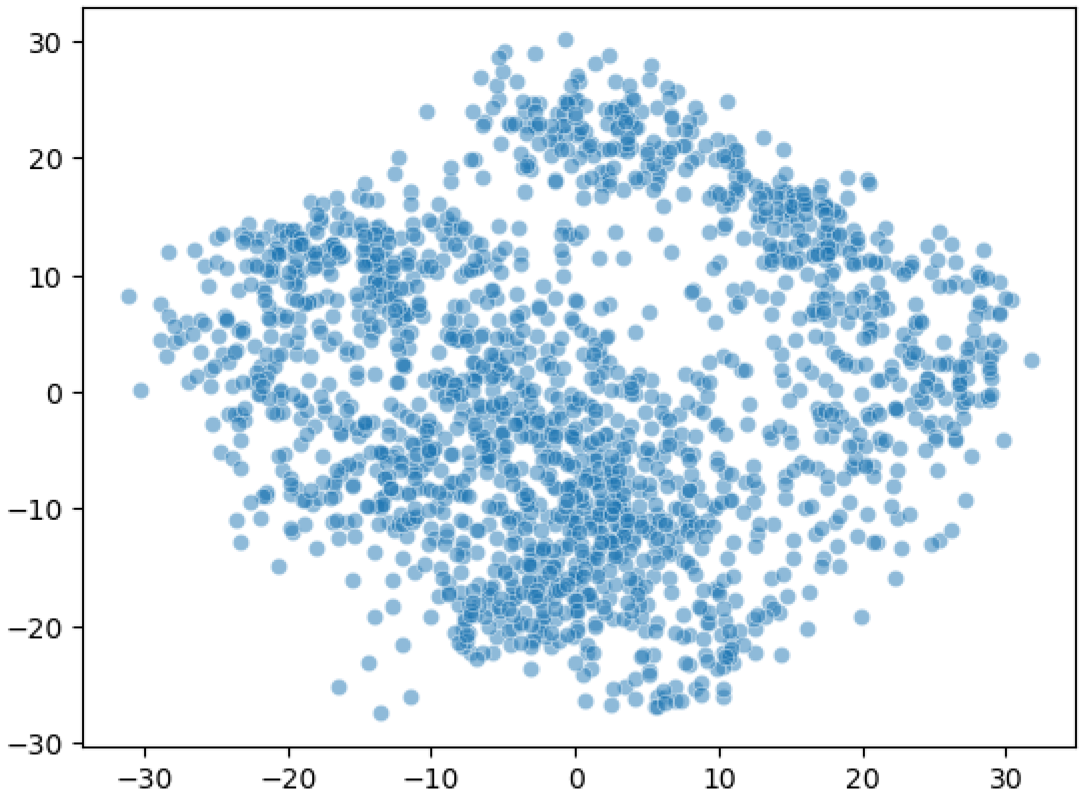
In [3]:
```
# We can use plt.imshow() to display one of the images as an example.
# 'digits['images']' is a list of images of size 8x8 pixels.
# We can plot the first image using plt.imshow with cmap=plt.cm.gray_r
# You should see a black '0' on a white background.
plt.imshow(digits['images'][0], cmap=plt.cm.gray_r);
```



In [4]:
```
# First we'll reduce our dataset from 64 to 2 dimensions using PCA for plotting

# Import PCA from sklearn
from sklearn.decomposition import PCA

# Instantiate a pca object that will result in 2 components being returned.
#   Use n_components=2, random_state=512
# Store as pca
pca = PCA(n_components=2, random_state=512)

# Transform X_digits to 2D using fit_transform.
# Store as X_2D
X_2D = pca.fit_transform(X_digits)

# assert that the dataset has been reduced to 2 dimensions (2 columns)
assert X_2D.shape[1] == 2
```

In [5]:
```
# Create a scatterplot of X_2D

# Use sns.scatterplot to plot X_2D with
#   the first column of X_2D on the x-axis
#   the second column of X_2D on the y-axis
#   alpha=.5 to make points slightly transparent
sns.scatterplot(x = X_2D[:,0], y = X_2D[:,1], alpha=.5)
```

Out[5]:
```
<AxesSubplot: >
```

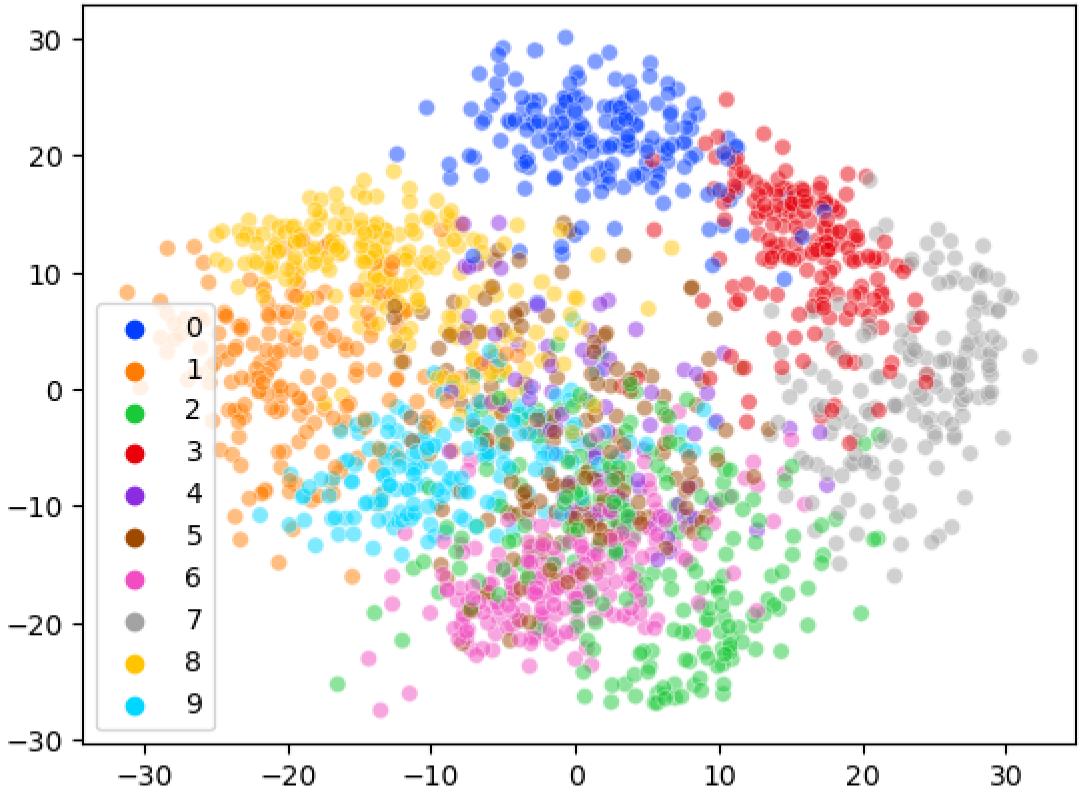


In [6]:
```
# Cluster the full dataset X_digits using KMeans

# Import KMeans from sklearn
from sklearn.cluster import KMeans

# Intantiate a KMeans object
#   which will generate 10 clusters
#   use init='k-means++'
#   and random_state=512
#   all other parameters as default (including init='k-means++')
# Store as km
km = KMeans(n_clusters = 10, init='k-means++', random_state=512)

# Use .fit_predict() on X_digits to both fit our k-means model and generate cluster assignments.
# Store the result as cluster_assignments_km
cluster_assignments_km = km.fit_predict(X_digits)

# print the first 15 cluster assignments in cluster_assignments_km
#   They should all be integers between 0 and 9 inclusive
cluster_assignments_km[:15]
```

Out[6]:
```
array([0, 2, 2, 1, 7, 8, 3, 6, 2, 8, 0, 4, 9, 1, 7], dtype=int32)
```

In [7]:
```
# Plot X_2D as a scatterplot colored by their cluster assignments

# Use sns.scatterplot to plot X_2D again with
#   the first column of X_2D on the x-axis
#   the second column of X_2D on the y-axis
#   alpha=.5 to make points slightly transparent
#   hue=cluster_assignments_km to color each point by it's cluster assignment
#   palette='bright' to use a brighter color palette than the seaborn default
sns.scatterplot(x = X_2D[:,0], y = X_2D[:,1], alpha=.5, hue=cluster_assignments_km, palette='bright')


# Note that the legend shows cluster assignments, not digit label
# Also note that, since we are clustering in 64 dimensional space, the clusters will
#   appear to overlap in 2 dimensional space
```

Out[7]:
```
<AxesSubplot: >
```



In [8]:
```
# What are the labels for images placed in cluster 9?

# Since we actually have labels for these images we can see which
#   digits were placed in cluster 9 without looking at the images themselves.

# Use y_digits and cluster_assignments_km to print the labels for
#   images assigned to cluster 9
# You should see that most of the labels are digit 2 with some 1s and 8s
y_digits[cluster_assignments_km == 9]
```

Out[8]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1,
       2, 1, 2, 1, 2, 3, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 1, 1, 1, 1, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 8, 2,
       2, 1, 2, 1, 2, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 8, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 8, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [9]:
```python
# Evaluate clustering using Homogeneity

# sklearn has a metric homogeneity_score which measures the homogeneity of clusters
# It returns a value between 0 and 1, where a higher value means more homogeneous.
# Compare the clustering assignments of Kmeans found above with an HAC model using Ward linkage
#
# Refer to the documentation here:
#   https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_score.html

# import homogeneity_score from sklearn.metrics
from sklearn.metrics import homogeneity_score

# import AgglomerativeClustering from sklearn.cluster
from sklearn.cluster import AgglomerativeClustering

# generate cluster assignments of X_digits with AgglomerativeClustering
#   using n_clusters=10
#   and linkage='ward'
#   all other parameters as default
# store as cluster_assignments_hac
cluster_assignments_hac = AgglomerativeClustering(n_clusters=10, linkage='ward').fit_predict(X_digits)

#Print the homogeneity scores for the clustering assigned by KMeans
print(f'homogeneity score for KMeans: {homogeneity_score(y_digits,cluster_assignments_km):0.2f}')

#Print the homeogeneity score for the clustering assinged by Agglomerative Clustering
print(f'homogeneity score for HAC    : {homogeneity_score(y_digits,cluster_assignments_hac):0.2f}')
```

```
homogeneity score for KMeans: 0.74
homogeneity score for HAC    : 0.86
```

In [ ]: