Michigan Tech

MA 5751: Statistical Data Mining

Analyzing Concrete Data

Unsupervised Learning Milestone W5

MA 5751 April 24, 2022

Lead: Ezequiel Carrillo, Assistant Lead: Christopher Barbour

Professor: Dr. Yeonwoo Rho

Table of Contents

1 Background

Concrete is a valuable resource used in nearly every infrastructure project. Given its high importance, the purpose of our analysis is to build models capable of taking in the input data and using the predictors to properly train and tune our models. In this portion of the project, the goal is to utilize our predictors to naturally separate our observations into clusters. This will allow our observations to group into sets with similar qualities/properties. For example, suppose concrete types are categorized into grades in the real world. Suppose now that a type A concrete is suitable for robust projects such as highways, airports, etc. Also suppose that a class D concrete is only suitable for a lightly transited driveway, wouldn't it be useful to separate our observations into these 'grades' such that we can see what category a certain observation falls in to? That in essence is the aim of the Unsupervised learning milestone. However, it is worth noting that a substantial amount of subject matter expertise is required in order to properly interpret and assess such clusters. My team and myself lacked this domain knowledge.

2 Variable Introduction and Definitions:

The data has been provided by Kaggle (https://www.kaggle.com/maajdl/yeh-concret-data).The response variable is a continuous variable which denotes the strength of a concrete sample using megapascals (csMPa). Below is a brief review of our predictors.

Predictor Descriptions:

Blast Furnace Slag: Blast-furnace slag is obtained by quenching molten iron slag from a blast furnace in water or steam, to produce a glassy, granular product that is then dried and ground into a fine powder.

Cement: Cement is a binder, a substance used for construction that sets, hardens, and adheres to other materials to bind them together.

Fly Ash: Fly ash is the fine ash produced at coal-fired power plants that develops cementitious properties when mixed with cement and water.

Water: Water is the key ingredient, which when mixed with cement, forms a paste that binds the aggregate together.
Superplasticizer: Superplasticizers (SPs), also known as high range water reducers, are additives used in making high strength concrete.
Coarse Aggregate: Coarse aggregates are a construction component made of rock quarried from ground deposits.

Fine Aggregate: Fine aggregates are essentially any natural sand particles won from the land through the mining process. Fine aggregates consist of natural sand or any crushed stone particles that are ¼" or smaller.

Age: Concrete strength increases with age as long as moisture and a favorable temperature are present for hydration of cement.

---

3 Data Pre-processing

Before we go about building the predictive models, we must pre-process the data by performing the appropriate transformations. For example, among the most commonly used are checking if the predictors lie on the same scale, checking for correlations between predictors, and investigating if there are predictors with near zero variance. For each of these issues, if present, we will resolve by performing a transformation on the data where appropriate. Below are a few records of our raw data prior to performing any data pre-processing. There were no missing values.

cement	slag	flyash	water	superplasticizer	coarseaggregate	fineaggregate	age	csMPa	Category
540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99	above
540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89	above
332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27	above
332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05	above
198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30	above
...	...	...	...	...	...	...	...	...	...
276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.28	above
322.2	0.0	115.6	196.0	10.4	817.9	813.4	28	31.18	below
148.5	139.4	108.6	192.7	6.1	892.4	780.0	28	23.70	below
159.1	186.7	0.0	175.6	11.3	989.6	788.9	28	32.77	below
260.9	100.5	78.3	200.6	8.6	864.5	761.5	28	32.40	below

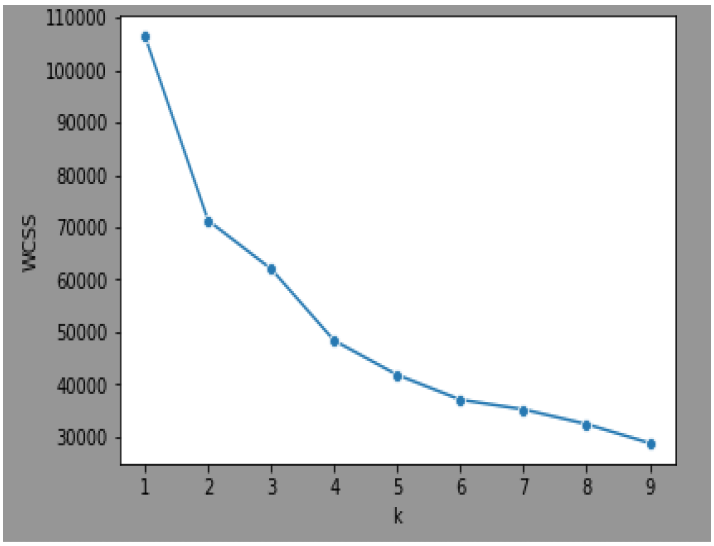
First of all, we removed the csMPa response variable and only used the original eight predictors to form the clusters. Secondly, given that our predictors did not lie on the same scale, data normalization was performed prior to applying any clustering algorithm on our dataset.

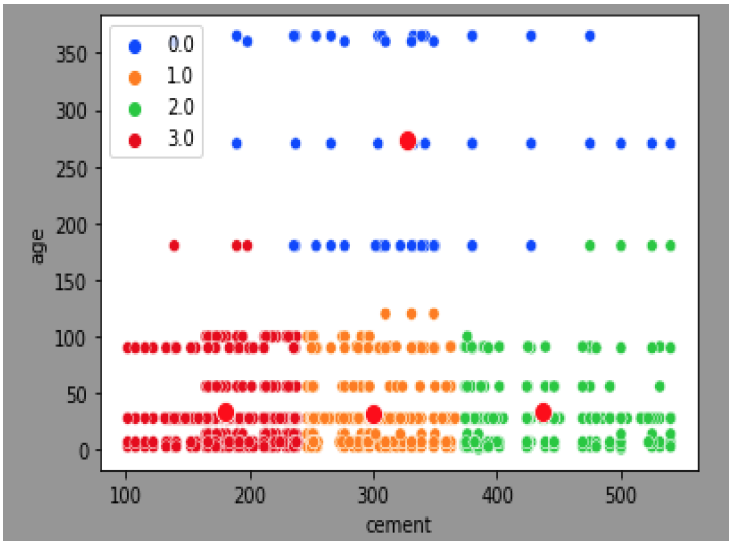
cement	slag	flyash	water	superplasticizer	coarseaggregate	fineaggregate	age
0.395564	0.000000	0.0	0.118669	0.001831	0.761826	0.495187	0.020511
0.392536	0.000000	0.0	0.117761	0.001817	0.766900	0.491397	0.020354
0.273422	0.117181	0.0	0.187489	0.000000	0.766403	0.488458	0.222027
0.268004	0.114859	0.0	0.183774	0.000000	0.751218	0.478781	0.294200
0.145460	0.096973	0.0	0.140626	0.000000	0.716605	0.604618	0.263673

4 Clustering

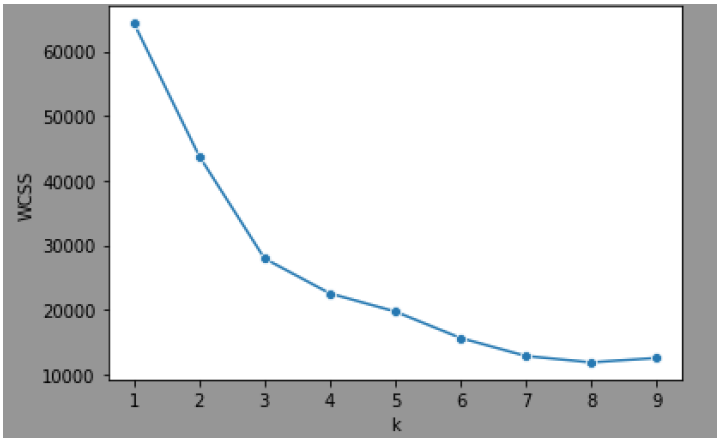
The type of clustering performed were K-Means Clustering and Hierarchical Clustering using ward's linkage and using Euclidean distance. Below are the clusters using variables which best separated the clusters.

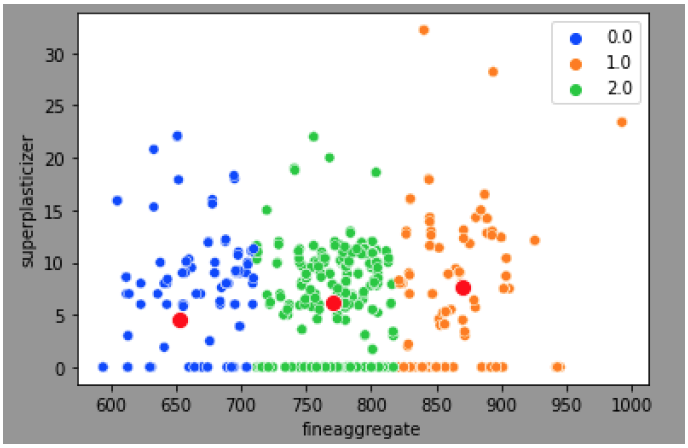
K Means, K = 4, x-axis: cement, y-axis: age



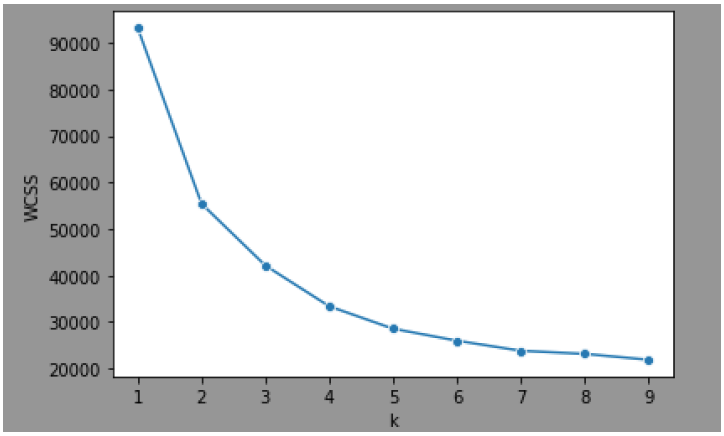


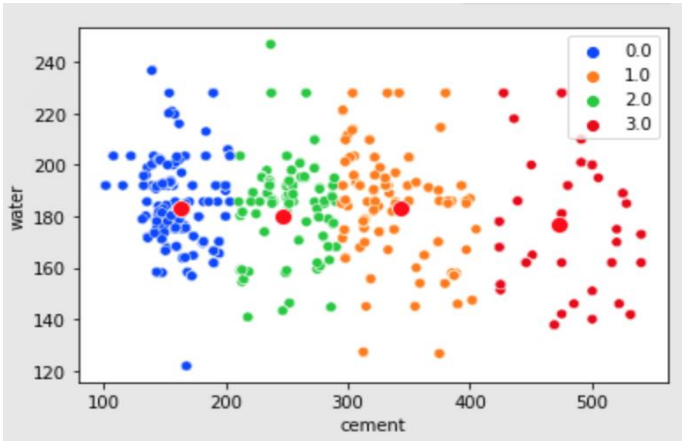
K Means, K = 3, x-axis: fineaggregate y-axis: superplasticizer



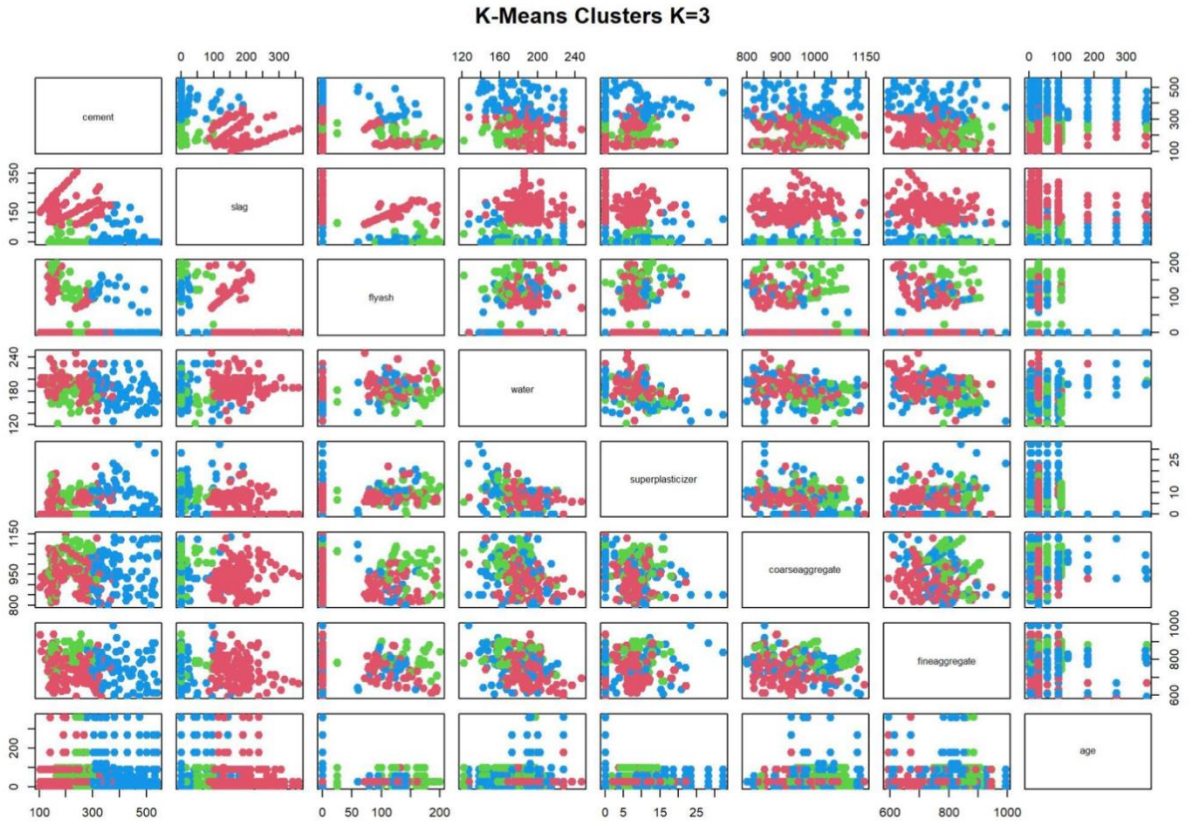


K Means, K = 4, x-axis: cement, y-axis: water



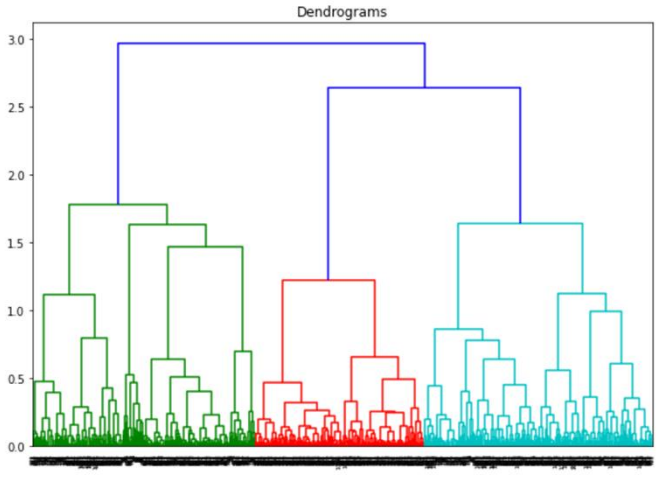


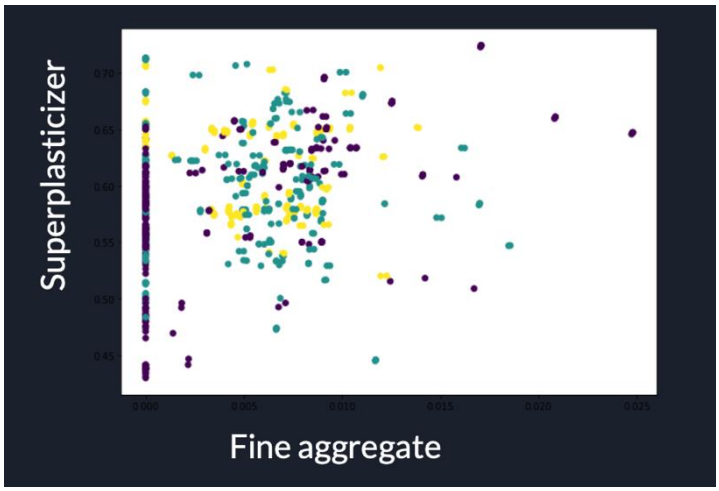
In summary, for K-Means the k = 3,4 were appropriate values for k. The below plot illustrates this as well.



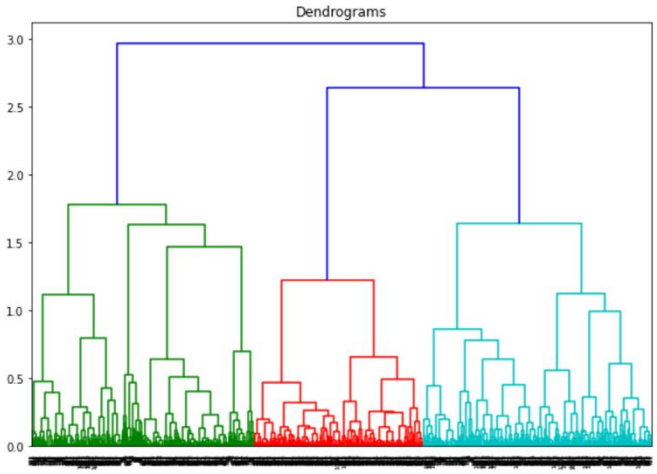
The first figure shows that choosing K=3 provides the most obvious clustering of different groups. Looking at 2nd figure, where we choose K=4, we still see many plots with dissimilar groups such as cement and fineaggregate that are similar to choosing K=3. Figure 3 shows that as we choose larger K values, the data does not seem to be able to cluster as well into dissimilar clusters. Choosing K=3 using K-Means clustering would be a good choice.

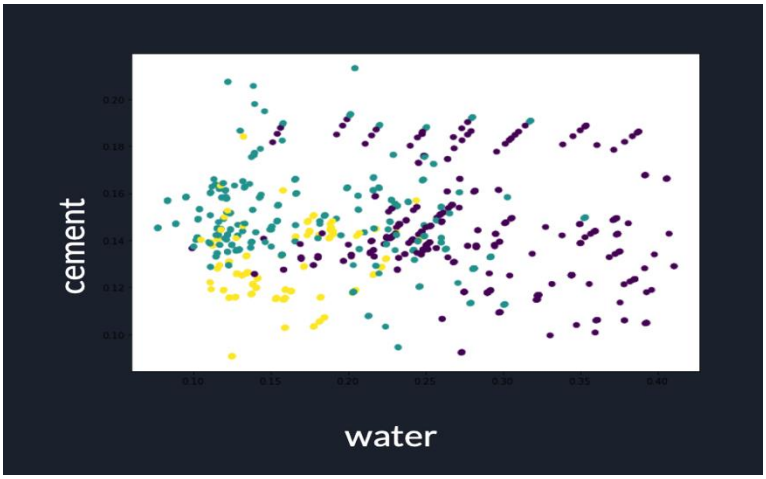
Hierarchical, c = 3, x-axis: superplasticizer, y-axis: age





Hierarchical, c = 3, x-axis: cement, y-axis: water





Inspecting the hierarchical clusters, this seems to confirm our previous claim, that the clustering method with our data seems to overlap. Even with selecting a maximum of 3 clusters, the second cluster has only two predictors that have a zero mean. K-means clustering appears to cluster the observations that are similar in a far superior manner than hierarchical.

Recall that the response variable was split into two parts to form a categorical response in W3. Given the binary output class structure, we cannot quantify the misclassification error because the number of clusters is greater than 2. However, from the plots, we find that the hierarchical clustering tends to have more overlap between clusters when compared to K-Means clustering. K-Means clustering tends to tightly “pack” the observations within clusters and his little to no overlap between observations.

5 Summary

K-Means seems to perform the cluster creations much better than hierarchical. Furthermore, it is worth noting some of the limitations and challenges we face. For example, given that we only have two classes from our categorical response variable, it is somewhat difficult to assess the performance as the data structure stands. One thing that would possibly be helpful would be to add another class such that each of our clusters has a corresponding label in the categorical response variable. This, however, presents its own challenges as this task requires domain knowledge in materials science.

Also, given the limited methods available for feature selection in an unsupervised learning setting, the model can be further improved by using both a qualitative and quantitative approach in selecting the variables. Furthermore, another limitation is that our clusters our being formed in a 2-D space, perhaps clustering in a higher dimensional space would more naturally group our observations, which would result in a much more improved model. In summary, we understand that our limited understanding in the subject matter limits our exploration capabilities in an unsupervised learning setting where there is no response variable, thus, a little bit more background understanding of the dataset context would be beneficial in improving our analysis.

6 Python Code:

```
# -*- coding: utf-8 -*-

"""Copy of Clustering W5.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1U_ajciRxvlfNIdGCtyZG8G-WGmbfvdVD
"""

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from google.colab import files
import cv2

concrete['csMPa'].describe()

from pandas.core.arrays import categorical
concrete = pd.read_csv('/content/Concrete.csv')
concrete.head()
categorical = pd.cut(concrete.csMPa,bins=[0,34,82],labels=['below','above'])
concrete.insert(9,'Category',categorical)
```

```
from sklearn.preprocessing import normalize

concrete = pd.read_csv('/content/Concrete.csv')

data_scaled = normalize(concrete)

data_scaled = pd.DataFrame(data_scaled, columns=concrete.columns)

data_scaled.head()


concrete


concrete.isnull().values.any()


def calculate_cost(X, centroids, cluster):
  sum = 0
  for i, val in enumerate(X):
    sum += np.sqrt((centroids[int(cluster[i]), 0]-val[0])**2 +(centroids[int(cluster[i]), 1]-val[1])**2)
  return sum


def kmeans(X, k, data):
  diff = 1
  cluster = np.zeros(X.shape[0])
  centroids = data.sample(n=k).values
  while diff:
    # for each observation
    for i, row in enumerate(X):
      mn_dist = float('inf')
      # dist of the point from all centroids
      for idx, centroid in enumerate(centroids):
        d = np.sqrt((centroid[0]-row[0])**2 + (centroid[1]-row[1])**2)
        # store closest centroid
        if mn_dist > d:
```

```
            mn_dist = d
            cluster[i] = idx
    new_centroids = pd.DataFrame(X).groupby(by=cluster).mean().values
    # if centroids are same then leave
    if np.count_nonzero(centroids-new_centroids) == 0:
      diff = 0
    else:
      centroids = new_centroids
  return centroids, cluster


conc = data_scaled.loc[:, ['csMPa', 'water']]
X = conc.values
sns.scatterplot(X[:,0], X[:, 1])
plt.xlabel('csMPa')
plt.ylabel('water')
plt.show()



cost_list = []
for k in range(1, 10):
  centroids, cluster = kmeans(X, k,conc)
  # WCSS (Within cluster sum of square)
  cost = calculate_cost(X, centroids, cluster)
  cost_list.append(cost)

sns.lineplot(x=range(1,10), y=cost_list, marker='o')
plt.xlabel('k')
plt.ylabel('WCSS')
plt.show()
```

```
k = 6

centroids, cluster = kmeans(X, k,conc)

sns.scatterplot(X[:,0], X[:, 1], hue=cluster,palette='bright')

sns.scatterplot(centroids[:,0], centroids[:, 1], s=100, color='r')

plt.xlabel('csMPa')

plt.ylabel('water')

plt.show()


conc = data_scaled.loc[:, ['water', 'slag']]

X = conc.values

sns.scatterplot(X[:,0], X[:, 1])

plt.xlabel('water')

plt.ylabel('slag')

plt.show()



cost_list = []

for k in range(1, 10):

  centroids, cluster = kmeans(X, k,conc)

  # WCSS (Within cluster sum of square)

  cost = calculate_cost(X, centroids, cluster)

  cost_list.append(cost)


sns.lineplot(x=range(1,10), y=cost_list, marker='o')

plt.xlabel('k')

plt.ylabel('WCSS')

plt.show()
```

```
k = 3

centroids, cluster = kmeans(X, k,conc)

sns.scatterplot(X[:,0], X[:, 1], hue=cluster,palette='bright')

sns.scatterplot(centroids[:,0], centroids[:, 1], s=100, color='r')

plt.xlabel('water')

plt.ylabel('slag')

plt.show()


conc = data_scaled.loc[:, ['cement', 'age']]

X = conc.values

sns.scatterplot(X[:,0], X[:, 1])

plt.xlabel('cement')

plt.ylabel('flyash')

plt.show()



cost_list = []

for k in range(1, 10):

  centroids, cluster = kmeans(X, k,conc)

  # WCSS (Within cluster sum of square)

  cost = calculate_cost(X, centroids, cluster)

  cost_list.append(cost)


sns.lineplot(x=range(1,10), y=cost_list, marker='o')

plt.xlabel('k')

plt.ylabel('WCSS')

plt.show()
```

```
k = 4

centroids, cluster = kmeans(X, k,conc)

sns.scatterplot(X[:,0], X[:, 1], hue=cluster,palette='bright')

sns.scatterplot(centroids[:,0], centroids[:, 1], s=100, color='r')

plt.xlabel('cement')

plt.ylabel('age')

plt.show()


import scipy.cluster.hierarchy as shc

plt.figure(figsize=(10, 7))

plt.title("Dendrograms")

dend = shc.dendrogram(shc.linkage(data_scaled, method='ward'))

from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='complete')

cluster.fit_predict(data_scaled)

plt.figure(figsize=(10, 7))

plt.scatter(data_scaled['csMPa'], data_scaled['water'], c=cluster.labels_)

from sklearn import metrics

labels = cluster.labels_

metrics.silhouette_score(data_scaled, labels, metric = 'euclidean')

plt.figure(figsize=(10, 7))

plt.scatter(data_scaled['water'], data_scaled['slag'], c=cluster.labels_)

plt.figure(figsize=(10, 7))

plt.scatter(data_scaled['cement'], data_scaled['age'], c=cluster.labels_)


plt.figure(figsize=(10, 7))

plt.scatter(data_scaled['cement'], data_scaled['coarseaggregate'], c=cluster.labels_)
```

7 Sources

Witten, D. M., & Tibshirani, R. (2010). A framework for feature selection in clustering. *Journal of the American Statistical Association*, *105*(490), 713–726. https://doi.org/10.1198/jasa.2010.tm09415

A. Angeli and A. Davison, "Live feature clustering in video using appearance and 3d geometry," in *Proc. BMVC*, 2010, pp. 41.1–11, doi:10.5244/C.24.41.

B. Zare, *ECE 8735 Notes Lecture 20, Hierarchical Clustering*. Uni- versity of Missouri, 2013

Poux, F., & Billen, R. (2019). Voxel-based 3D point cloud semantic segmentation: unsupervised geometric and relationship featuring vs deep learning methods. *ISPRS International Journal of Geo-Information*. 8(5), 213; https://doi.org/10.3390/ijgi8050213