Obesity Categorization Clustering around Various Traits

Unsupervised Learning

Cody Rorick

MA 5751

Fall 2024

Obesity is a problem that many Americans are struggling with today. There is still some disagreement however on why exactly this is the case and what traits/lifestyle choices contribute to obesity the most. For example, two articles from highly reputable sources that were considered in the motivation for this analysis take opposing stances on the relationship between calorie intake and weight gain. One article takes the stance that unhealthy weight is heavily influenced by the equation of calories in vs. calories burned, the other article takes the stance that not all calories are equal and that calories alone do not tell the whole story. The dataset gathered here contains information that may be related to obesity, and different categorizations of obesity such as Insufficient_Weight, Normal_Weight, Obesity_Type_I, Obesity_Type_II, Obesity_Type_III, Overweight_Level_I, and Overweight_Level_II. An analysis will perform different unsupervised learning techniques such as PCA, k-means clustering, and hierarchical clustering to see if the predictors gathered do a good job of detecting and naturally grouping people into their respective obesity categorization. Do certain traits tend to naturally group people into obesity and non obesity, or do people randomly fall into their categorization regardless of the traits collected in this dataset? Are there patterns that distinguish obesity types, such as obesity type I and obesity type III or is there not a distinguishable difference in these groups? That is what this analysis intends to look into further.

Table of Contents

Introduction

One way to get a better understanding of traits that are often found in people with different categorizations of obesity is to collect data that is theorized to be contributing factors and see how they naturally cluster together with unsupervised learning, leaving out hypotheses about direct predictors and responses from the analysis.

If certain predictor markers tend to cluster around different types of obesity, then it can be useful for further investigation into what traits commonly fall into the unhealthy obesity levels and what traits commonly fall within healthier obesity levels. If these lifestyle variables are good at separating out which obesity level a person fits at, then this will allow us to draw some conclusions about general lifestyle habits that are associated with different obesity levels.

If the clusters do not group well around the various obesity types, then it is an indicator that there is overlap between different obesity categorizations with respect to these traits and behaviors, and perhaps different indicators may be better at separating people into their respective obesity diagnosis.

Principal Component Analysis (PCA)

Principal component analysis assists in getting a sense of how variation falls within a dataset. Analyzing the data in its raw and unaltered form gives a sense of the magnitude of the predictors and their variability in their natural scale of measurement in comparison to other predictors in the data. Analyzing the data with PCA once it has been normalized can give a sense of how much the data varies within its own scale regardless of magnitude. The natural variance and magnitudes no longer play a role once the data has been normalized. One of the biggest advantages of normalizing the data is that it can tell you how much variability it contains in direct comparison to other types of data that are measured on different scales.

The first step in understanding the obesity data was to run a PCA on the unscaled data. This was an attempt to see how much of the variability in the raw data was dispersed across the different variables collected.

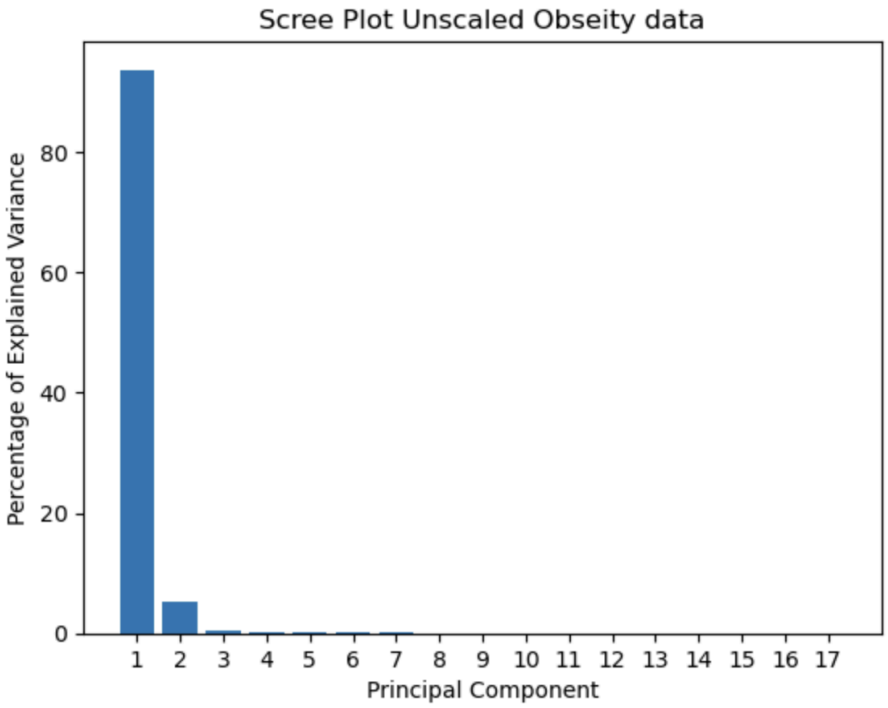

Figure 1 Unscaled PCA on Obesity Data

As can be seen above, the variability is heavily concentrated in the raw data to one component. This is an indicator that only a small number of predictors explain most of the raw variability. If the variation in the overall data was due to more predictors, more components would be shown as influential in explaining the variation of the data. To see which variables were dominating the variability the most, a feature importance plot was constructed.

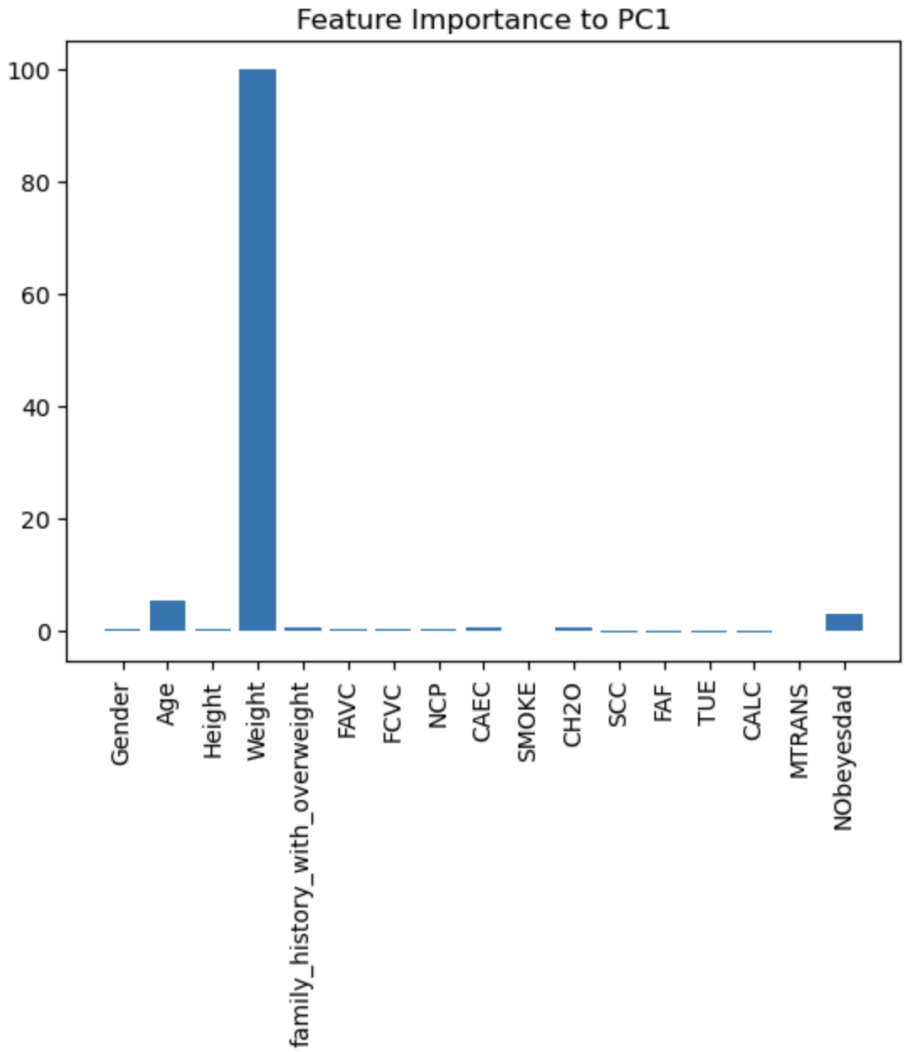

Figure 2 Unscaled Feature Importance

The most important variable in determining the variance of the data by a wide margin is weight. Age is the second most influential predictor in the dataset. Keeping this in mind, any further clustering analysis with raw data will heavily reflect the effects of weight on how the clusters get clumped together. To get a better sense of the variability in the data without any influence by the scale of the predictors, centering and scaling was performed and principal component analysis was reassessed.

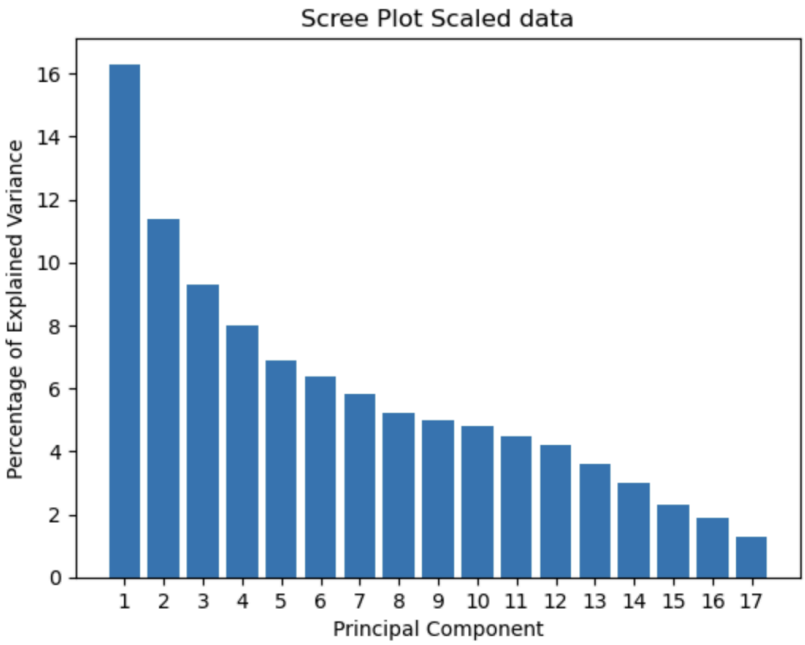

Figure 3 Normalized PCA

The centered and scaled data was shown to have a much more equitable dispersion of variance in the dataset. The variance was no longer heavily concentrated to the first component such as in the raw data. Now it can be anticipated that more variables in the original data will play an important role in explaining the variation. To confirm this, a feature importance plot was constructed for just the first principal component.

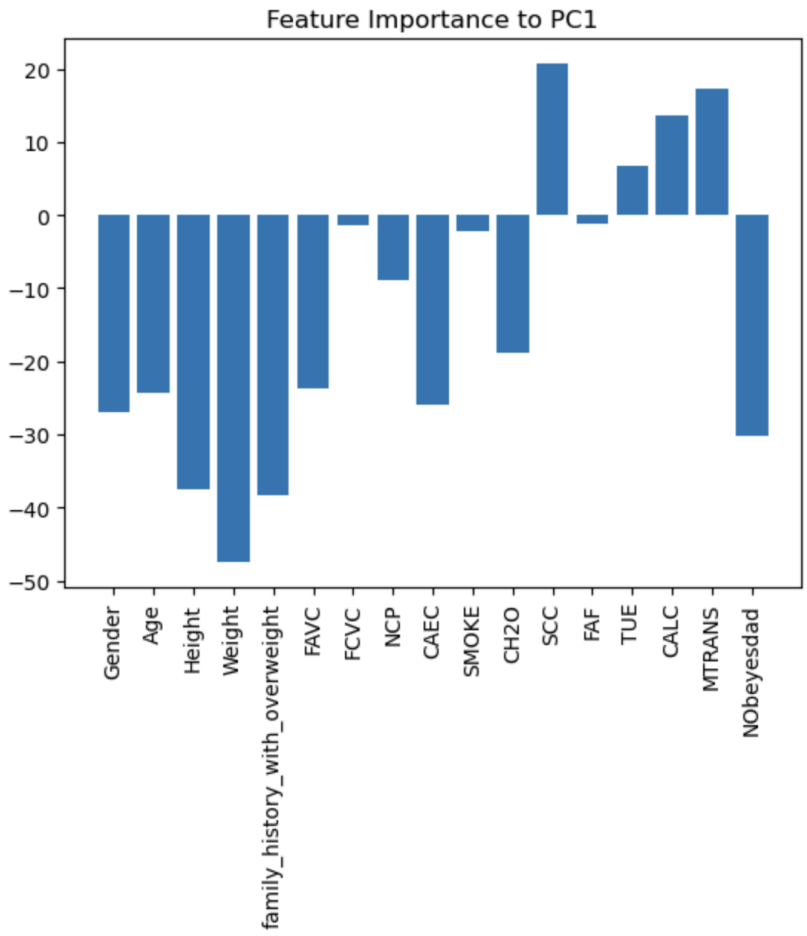

Figure 4 Scaled PCA Feature Loadings

As shown above, PC1 now has much greater variation in the loadings for each predictor. It can be seen that weight still contributes the most to PC1 variability, but other features are shown to also play a significant role. One interesting phenomenon is the flipping of signs on some of the feature contributions. This can indicate inverse effects on the variability. For example, a larger SCC variable (a variable which monitors calories) might be associated in future groupings with lower weight and vice versa. These loadings however only account for one principal component that explains about 16% of the variation in the data, further analysis on the other principal components would need to be done to ensure that this pattern holds.

Clustering

Clustering refers to a family of algorithms that groups data together naturally based on patterns found in variables. If certain variables are found in high quantities in one group and low quantities in another group for example, then the goal of a clustering algorithm is to be able to identify these as separate groups or clusters.

To understand the following sections better, it is best to know which obesity categories pertained to which numbered categories in the analysis. 0 was used to indicate Insufficient_Weight, 1 was used to indicate Normal_Weight, 2 was used to indicate Obesity_Type_I, 3 was used to indicate Obesity_Type_II, 4 was used to indicate Obesity_Type_III, 5 was used to indicate Overweight_Level_I, and 6 was used to indicate Overweight_Level_II.

K Clustering

Any K clustering technique requires that the number of k clusters are defined ahead of time. This can be any number less than the sample size of the data. Since I am trying to see if the obesity categorizations here will naturally cluster together, I am defining the number of clusters as 7, as that is how many obesity types are used in this dataset.

K-Means

The K means algorithm randomly places a predefined “k” number of points in the data space, calculates the distance between each of the individual points and these k points, puts the point in a group that is closest in distance to one of the k points, re calculates where that kth point should be placed based on the mean of its group, and then iterates until each of the k points stop moving.

The K-means algorithm was used on this data and the results for this are explored more below. First in this exploration is a table indicating the purity and a guess on the category that the kth group was attempting to cluster. The data used here was unscaled.

Table 1 Unscaled K-Means

Cluster	Most common category within cluster	Purity (%)
0	6	33.2
1	3	63.2
2	0	70.09
3	6	42.05
4	4	89.40
5	2	46.23
6	1	45.80

The table above shows some results from unscaled k-means clustering. The most common category indicates which actual category from the obesity data showed up the most in that cluster. The purity of the grouping in each category indicated how much that categorization appeared in comparison to the other categories. Some insights that can be gathered from this include the especially high purity of category 4, and category 5 not showing up most frequently

in any of the clusters. To get a better sense of how the data got grouped, histograms were created to show the category frequency within clusters.

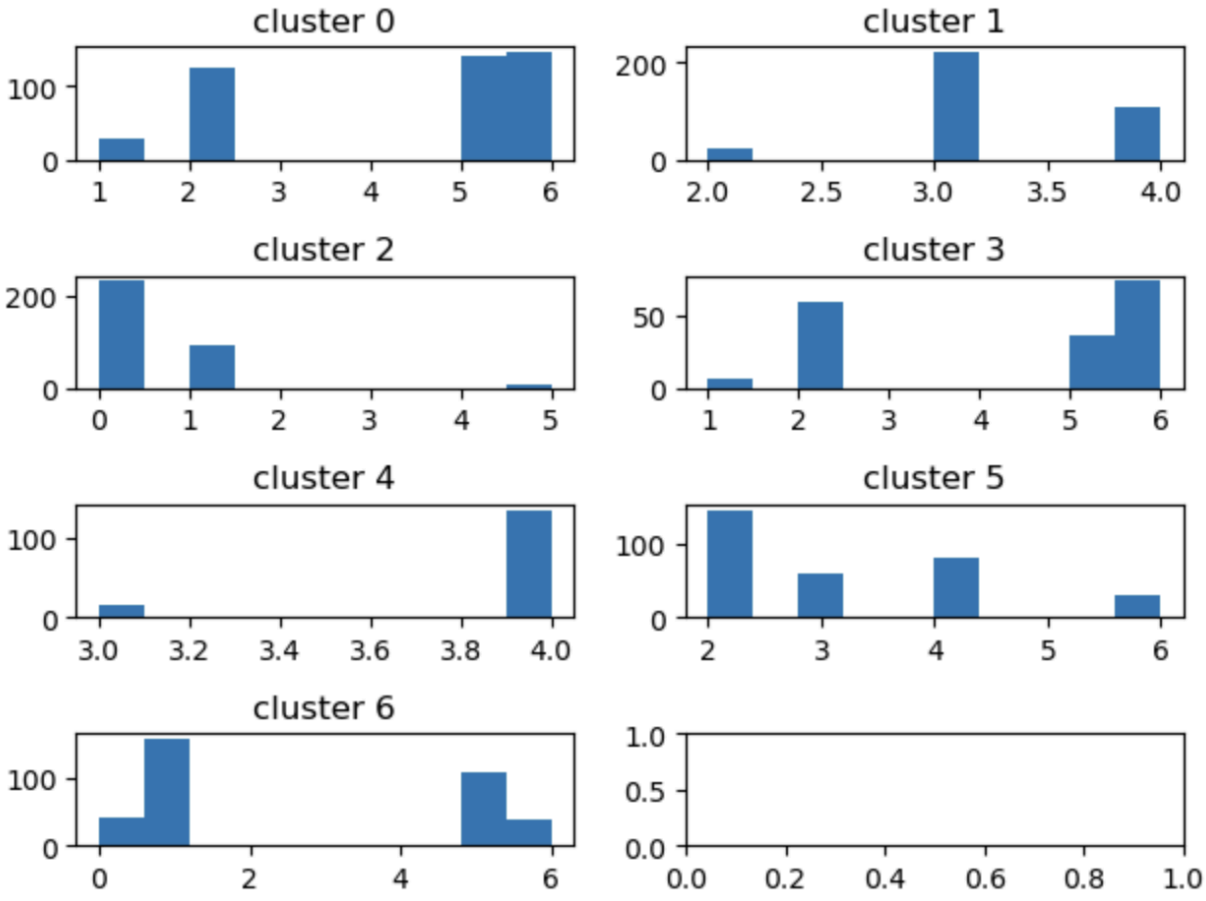

Figure 5 Unscaled K Means Clustering

This histogram above can be broken down further to explore some trends. Categories 5 and 6 competed for cluster 0, as they both showed up in high quantities. It can be seen that category 4 dominated cluster 4 and category 3 mostly dominated cluster 1 by themselves. Categories 2, 5, and 6, showed up together in high quantities in cluster 0, which is likely due to the similarities in weight between these groupings. A scaled approach was taken next to examine if these trends still exist without the large influence that weight's unscaled variance plays.

Table 2 Scaled K-Means Clustering

Cluster	Most common category within cluster	Purity
0	4	86.29
1	0	45.73
2	2	34.55
3	3	34.09
4	3	26.89
5	3	34.44
6	2	30.59

The purity was lower for most of the groups, however category 4 still remained clustered very well. Remember that category 4 contains obesity of type III, which appears to have certain characteristics that allow it to naturally differentiate itself from other groups. Groups 5 and 6 did not show up most frequently in any of the clusters that were created, indicating that the traits collected do not allow these groups to differentiate themselves from the other groups very well. The frequency distribution histograms are examined next to gain insight into how well the groupings were structured.

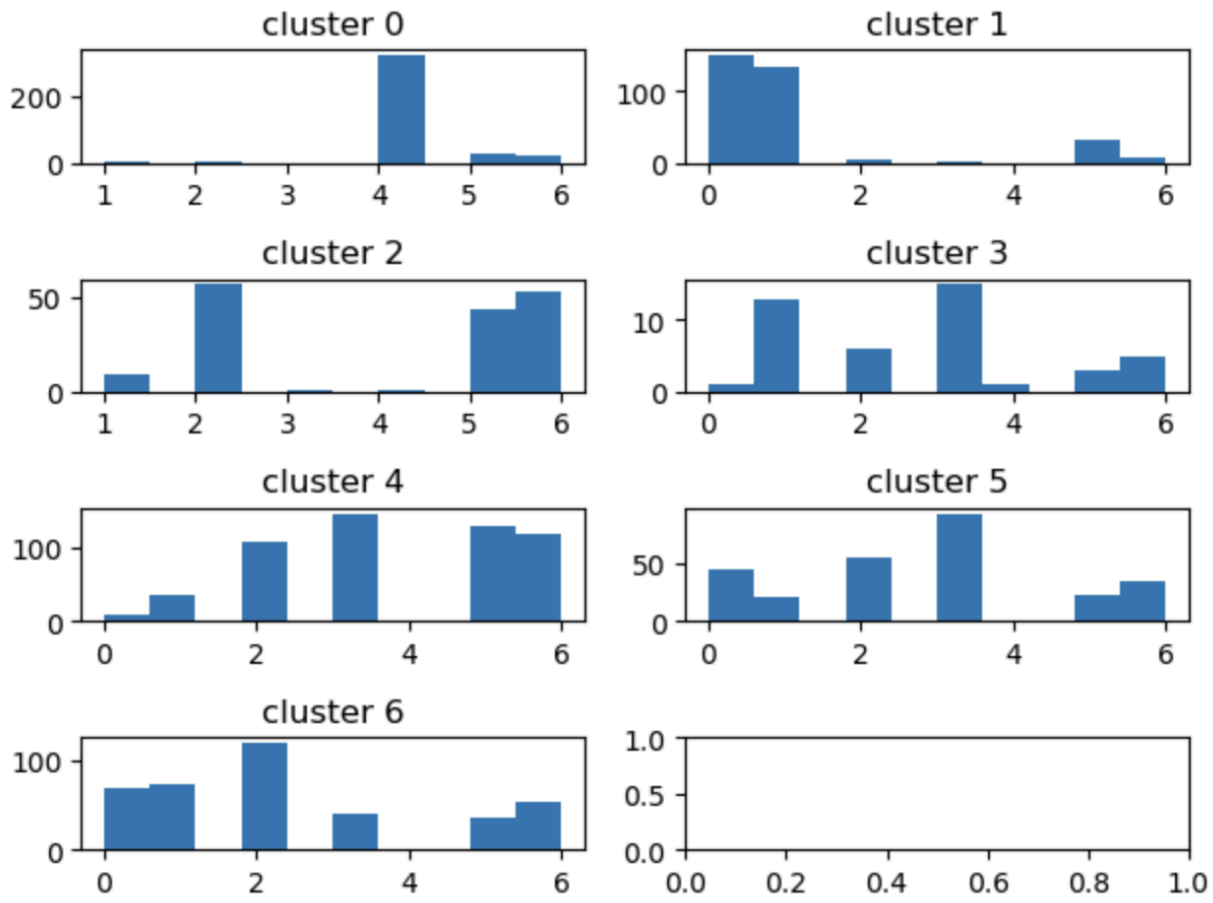

Figure 6 Normalized K-Means Clustering

The histogram above contains a lot of information on how the scaled data got grouped. Cluster 0 shows a strong presence of category 4 as was mentioned above, and none of the other categories have a substantial presence in that cluster. Category 4 was not commonly grouped into any of the other clusters, indicating that it doesn't share very many traits with the other groups. This would indicate that category 4 is a very unique obesity categorization. The clustering also shows that categories 0 and 1 could share similar attributes, as they both show up in very high quantities together in clusters 1 and 6.

K-Medoids

K-medoid clustering is a very similar algorithm to K-means clustering. The major difference is that instead of finding a centroid based on the calculation of a group mean, an actual point is chosen within a cluster to be the representative of that grouping's mean. Below, the efficacy of k-medoid clustering is analyzed.

Shown here is a table that examines some statistics from the k-medoid clustering approach on unscaled data.

Table 3 K-Medoid Clustering on Unscaled Data

Cluster	Most common category within cluster	Purity
0	3	54.06
1	6	36.13
2	4	41.15
3	2	44.53
4	1	46.18
5	0	69.76
6	6	43.09

The purity shown in this table for the different clusters is very similar to K-means. The K-medoid approach, similar to the k-means approach, had a difficult time differentiating category 5. Category 5 does not show up as the most common category in any of the clusters. One noticeable difference between the unscaled k-means clustering and the unscaled k-medoid clustering is the ability to cluster around category 4. The purity of cluster 2, the one in which category 4 is predicted the most, is much lower than the purity that k-means was able to achieve. The histogram below shows the frequency distribution of the categories in each cluster.

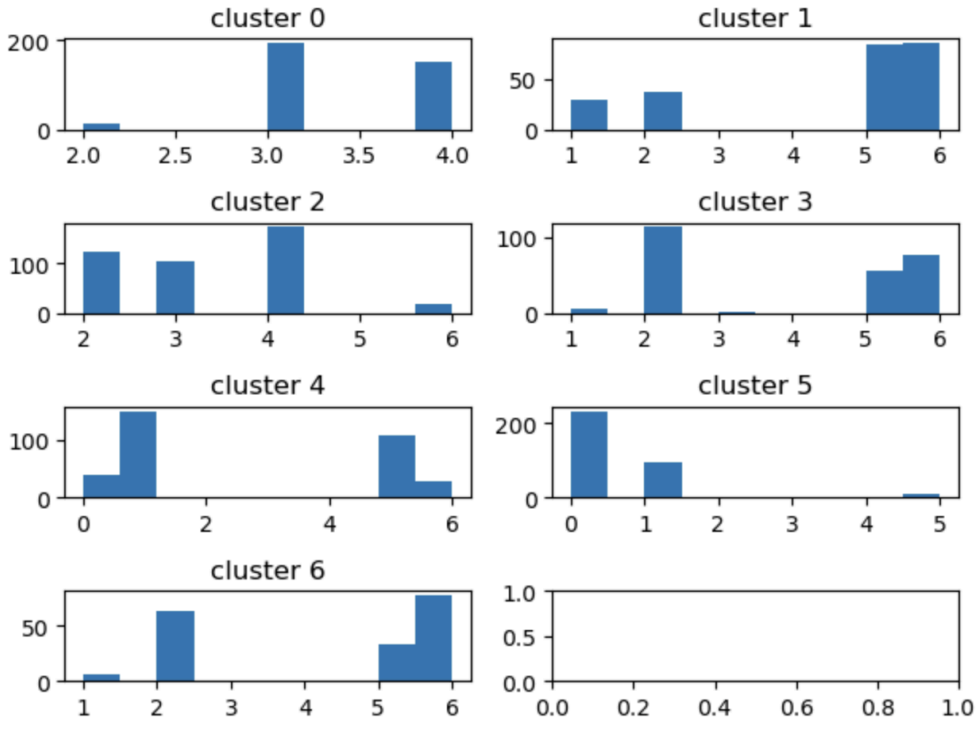

Figure 7 Scaled K-Medoid Clustering

The distributions shown here share a close semblance to the distributions drawn from the k-means analysis of raw data. As alluded to in the table above, there is a noticeable difference in the distribution of categories in cluster 2 where category 4 is the most common within the group, categories 2 and 3 are also showing up in high quantities. Category 4 was clustered with high purity using k-means. This indicates that the algorithm is having a harder time differentiating category 4 from other categories than the k-means approach did. Next, k-medoid will be used to cluster scaled data.

Table 4 Normalized K-Medoid Clustering

Cluster	Most common category within cluster	Purity
0	2	31.34
1	3	25.12
2	0	50.69
3	3	33.04
4	3	37.09
5	4	50.25
6	3	42.04

The purity of the scaled data is much lower than the unscaled data for k-medoid clustering. One noticeable difference from the k-means algorithm is that category 3 showed up on four separate occasions here, while it showed up only 3 times while using the k-means algorithm. This is a strong indication that category 3 has a lot of variation within its own group for the traits analyzed. The frequency of categories showing up in each cluster of the scaled data using the k-medoid algorithm is shown below.

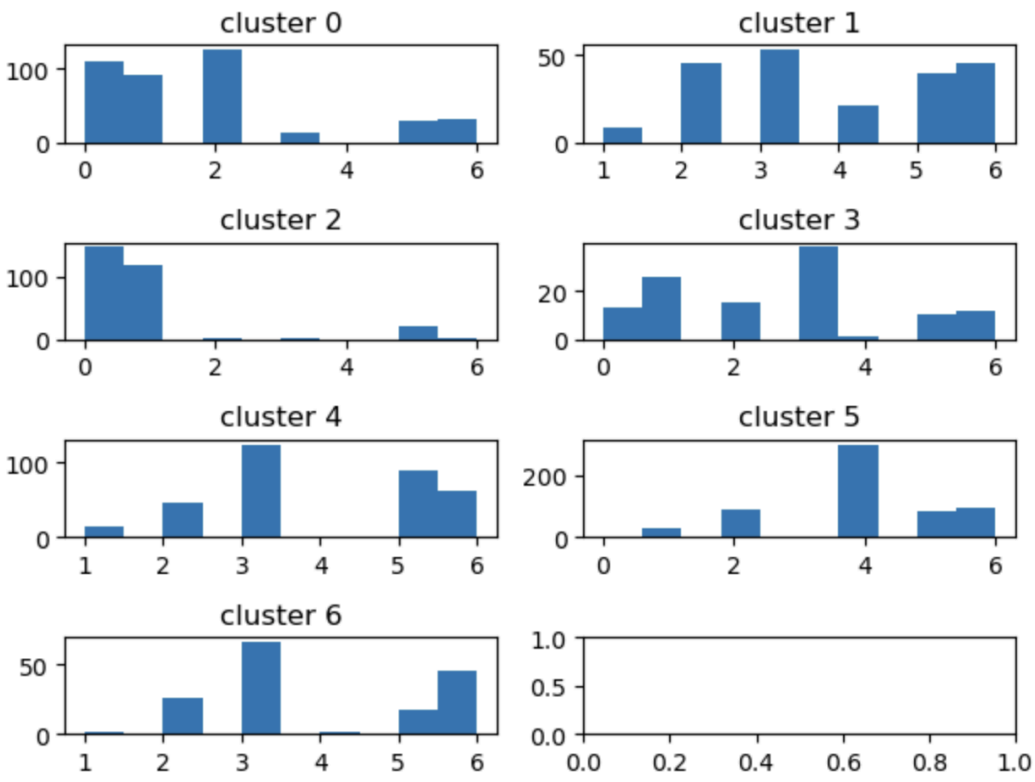

Figure 8 Normalized K-Medoid Clustering

This histograms again looks more dispersed than when utilizing the unscaled data, but some patterns can still be deduced. As the table suggests, category 3 - obesity type II, shows up in very high quantities amongst multiple different clusters. This is an indication that within this obesity categorization, there are many different strongly clustered patterns of lifestyle choices. It can also be seen in clusters 0 and 2 that categories 0 and 1 show up together in high quantities for both, which further confirms that these categories like to cluster together and share similarities.

Hierarchical Clustering

Hierarchical clustering techniques, unlike k clustering techniques, do not require the number of clusters to be defined before the analysis takes place. Each observation starts as its own grouping, and the groupings are combined with other groupings that are the most similar to it. This section uses euclidean distances and complete linkage to determine how similar groupings are to each other. Euclidean distance is the sum of squared difference between values. Complete linkage refers to a technique that finds the minimum euclidean distance between the furthest distances in a group and combining groups based on that.

To get a better interpretation of how each of the categorizations fell on the dendrograms, I placed the number indicator of the category below each node.

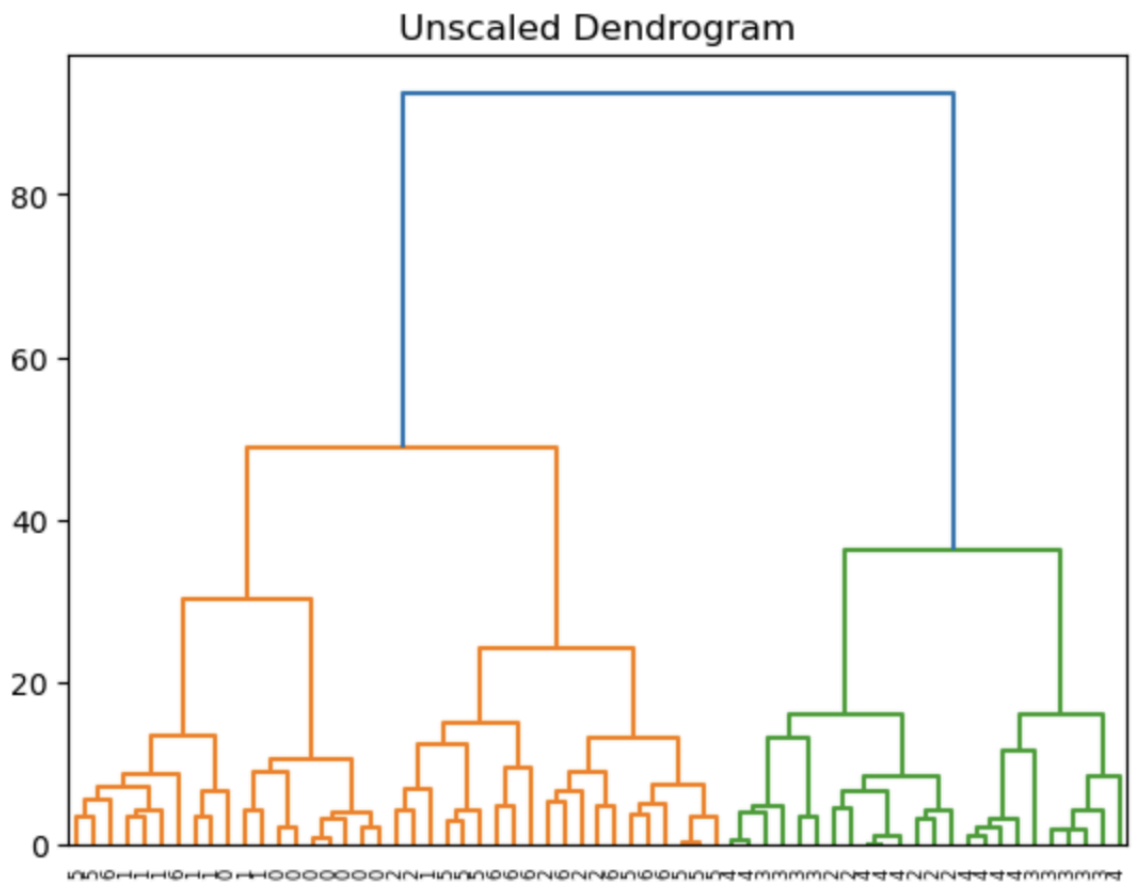

Figure 9 Unscaled Hierarchical Clustering

The unscaled dendrogram does show some mixing and matching amongst the numbered categorizations, but long strings of numbers are shown together. For example, the 0's here are shown mostly clumped together in the same branch on the left side of the dendrogram. This is a good indication that the algorithm is detecting similarities in that categorization. One very discernable pattern shown is the concentration of 2's, 4's, and 3's on the right side while 0's, 1's, 5's, and 6's are concentrated on the left side. This corresponds to the obesity types being clustered together on the right, and the non-obesity types clustered together on the left.

To see if patterns still exist without the large effects of weight variability, the data was centered and scaled to place equal weighting on each predictor. The results for this approach are shown below.

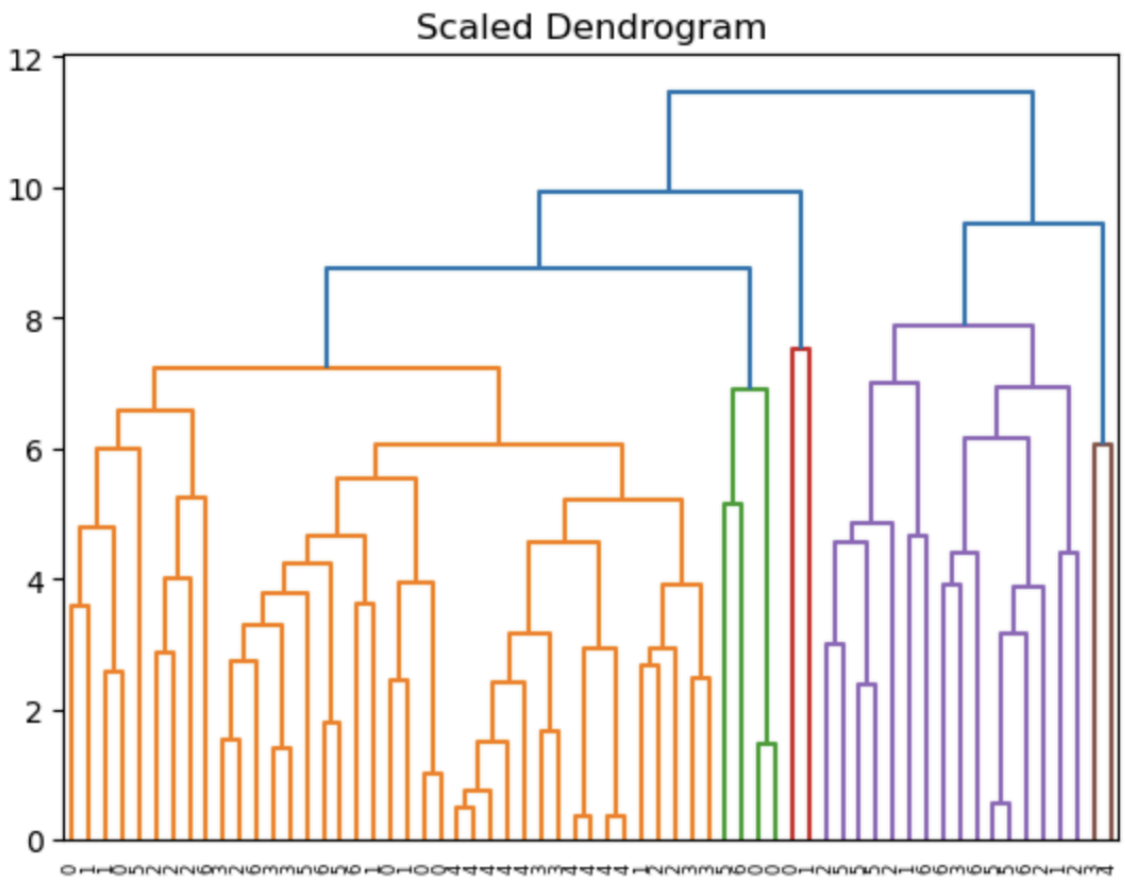

Figure 10 Normalized Hierarchical Clustering

The dendrogram above displays a much more random placement of obesity categorizations than the raw data did. The clustering does not do as good of a job at separating out obesity categorizations. One interesting exception to this are the 4 categorizations, they are very well grouped compared to the other categorizations. 4 corresponds to Obesity Type III, so even when the data is normalized, it appears that people with Obesity Type III share very similar characteristics. This was also suggested by the K-means and K-medoid clustering analysis done in prior sections.

Conclusion

This analysis has shown that the data tends to structure its clustering around obesity categorizations fairly well when the dataset was left unchanged. This is likely due to the large influence that weight has on obesity categorization. Allowing this term to dominate the predictors because it has a much larger variance and magnitude compared to the other terms in the data gives the clustering techniques a better chance at properly categorizing the data. If the goal is simply to cluster people into their respective obesity categorizations with the highest degree of accuracy, leaving the raw weight data in could prove to be advantageous. This analysis therefore shows how important the role a person's weight plays in determining what level of obesity they are diagnosed at, which is not a new insight.

The obesity groupings are shown to be much more random when the data has been normalized to control for the large variance of weight. There are some exceptions, for example the Obesity type III categorization was still shown to cluster very well. There were other useful insights from the normalized clustering too, such as that people categorized under insufficient weight and normal weight tended to cluster together, and that people categorized as having type II obesity showed very divisive clustering. Suggested further analysis based on these findings could include examining what traits are so strongly differentiating people with Obesity type III from other types of obesity, what traits are commonly shared between people of insufficient weight and normal weight, and what traits are causing the people with type II obesity to form strongly clustered subgroups.

This analysis showed why there may still be some disagreement in the literature as to what specific lifestyle indicators differentiate people with different types of obesity and non obesity the most. The unsupervised approach to examining the data still proved to be an excellent starting point that could drive future analysis, using additional unsupervised or supervised learning techniques to assist in understanding the data further.

Sources

ASN: Scientists Claim that Overeating Is Not the Primary Cause of Obesity.
https://nutrition.org/scientists-claim-that-overeating-is-not-the-primary-cause-of-obesity/

Harvard Health: Why people become overweight.
https://www.health.harvard.edu/staying-healthy/why-people-become-overweight

An Introduction to Statistical Learning with Applications in R, Second Edition
Authors: Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani

UC Irvine: Estimation of Obesity Levels Based On Eating Habits and Physical Condition.
https://archive.ics.uci.edu/dataset/544/estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition

Python Code

```
#importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from scipy.cluster.hierarchy import dendrogram, linkage, leaves_list
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from statistics import mode
from sklearn_extra.cluster import KMedoids

# Load the dataset in
file_path = 'ObesityDataSet_raw_and_data_sinthetic.csv'
data = pd.read_csv(file_path)

# Encode categorical variables via one hot encoding
data_encoded = data.copy()
for column in data_encoded.select_dtypes(include='object').columns:
    data_encoded[column] = LabelEncoder().fit_transform(data_encoded[column])
#Fitting a principal component analysis on Unscaled data
pca = PCA()
pca.fit(data_encoded)
pca_data = pca.transform(data_encoded)

# Grabbing the variance explained for each feature per PCA component
per_var = np.round(pca.explained_variance_ratio_* 100, decimals=1)
labels = [str(x) for x in range(1, len(per_var)+1)]
```

```
# Barplot to visualize variance explained by each principal component
plt.bar(x=range(1,len(per_var)+1), height=per_var, tick_label=labels)
plt.ylabel('Percentage of Explained Variance')
plt.xlabel('Principal Component')
plt.title('Scree Plot Unscaled Obseity data')
plt.show()

#Plotting the amount each variable contributes to the first component of PCA
feature_contributions = np.round(pca.components_[0]*100,1)
plt.bar(data_encoded.columns, feature_contributions)
plt.xticks(rotation = 'vertical')
plt.title('Feature Importance to PC1')
plt.show()
# Now centering and scaling the data for PCA
scaler = StandardScaler()
obesity_scaled = pd.DataFrame(scaler.fit_transform(data_encoded),
columns=data_encoded.columns)

#Fitting a principal component analysis on the centered and scaled data
pca = PCA()
pca.fit(obesity_scaled)
pca_data = pca.transform(obesity_scaled)

# Barplot to visualize variance explained by each principal component
plt.bar(x=range(1,len(per_var)+1), height=per_var, tick_label=labels)
plt.ylabel('Percentage of Explained Variance')
plt.xlabel('Principal Component')
plt.title('Scree Plot Scaled data')
plt.show()

#Plotting the amount each variable contributes to the first component of PCA
feature_contributions = np.round(pca.components_[0]*100,1)
plt.bar(data_encoded.columns, feature_contributions)
plt.xticks(rotation = 'vertical')
plt.title('Feature Importance to PC1')
plt.show()

# Agglomerative approach, in order to interpret dendograms
# a random stratified sample is taken
np.random.seed(500)
sample, _ = train_test_split(data_encoded, test_size=0.97,
stratify=data_encoded['NObeyesdad'])
sample = sample.reset_index(drop = True)
```

```
labelList = list(sample.loc[:,'NObeyesdad'])

# Dendogram for the unscaled data
linkage_data = linkage(sample, method='complete', metric='euclidean')
dendrogram(linkage_data, labels=labelList, distance_sort='descending')
plt.title('Unscaled Dendrogram')
plt.show()

#scaling data
scaler = StandardScaler()
obesity_scaled_sample = pd.DataFrame(scaler.fit_transform(sample),
columns=data_encoded.columns)

#plotting scaled data
linkage_data = linkage(obesity_scaled_sample, method='complete', metric='euclidean')
dendrogram(linkage_data, labels=labelList, distance_sort='descending')
plt.title('Scaled Dendrogram')
plt.show()

# K-means clustering. 7 clusters for 7 different obesity levels
kmeans = KMeans(n_clusters=7, random_state=42)
kmeans_labels = kmeans.fit_predict(data_encoded)

# K-medioid clustering. 7 clusters for 7 different obesity levels
kmedoids = KMedoids(n_clusters=7, random_state=42)
kmedoids_labels = kmedoids.fit_predict(data_encoded)

# Combining the "predictions" from the kmeans and kmedioid approaches
cluster_map = pd.DataFrame()
cluster_map['data_index'] = data_encoded.index.values
cluster_map['cluster_kmeans_unscaled'] = kmeans_labels
cluster_map['cluster_kmedioids_unscaled'] = kmedoids_labels

#Combining the predictions with the origninal obestiy data
obesity_clustering = pd.concat([data_encoded, cluster_map], axis = 1)

#What is the prediction for kmeans clustering?
for i in set(obesity_clustering['NObeyesdad']):
    print(f"The kmeans unscaled prediction for cluster {i} is group
        {mode(obesity_clustering.loc[obesity_clustering['cluster_kmeans_unscaled']==i,'NObeye
        sdad'])}")
    categories_in_cluster =
        list(obesity_clustering.loc[obesity_clustering['cluster_kmeans_unscaled']==i,'NObeyesda
        d'])
```

```
print(f"The purity of cluster {i} is
    {categories_in_cluster.count(mode(obesity_clustering.loc[obesity_clustering['cluster_km
    eans_unscaled']==i,'NObeyesdad'])) / len(categories_in_cluster)}")
```

```
#What is the prediction for kmedioid clustering?
for i in set(obesity_clustering['NObeyesdad']):
    print(f"The k medioids unscaled prediction for cluster {i} is group
{mode(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_unscaled']==i,'NObeyesdad'
])}")
    categories_in_cluster =
list(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_unscaled']==i,'NObeyesdad'])
    print(f"the purity of cluster {i} is
        {categories_in_cluster.count(mode(obesity_clustering.loc[obesity_clustering['cluster_km
        edioids_unscaled']==i,'NObeyesdad'])) / len(categories_in_cluster)}")

figure, axis = plt.subplots(4, 2)

# the categories that fall within group 0 for kmeans
axis[0,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_unscaled']==0,'NObeyesdad'])
axis[0, 0].set_title('cluster 0')

# the categories that fall within group 1 for kmeans
axis[0,
1].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_unscaled']==1,'NObeyesdad'])
axis[0, 1].set_title('cluster 1')

# the categories that fall within group 2 for kmeans
axis[1,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_unscaled']==2,'NObeyesdad'])
axis[1, 0].set_title('cluster 2')

# the categories that fall within group 3 for kmeans
axis[1,
1].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_unscaled']==3,'NObeyesdad'])
axis[1, 1].set_title('cluster 3')

# the categories that fall within group 4 for kmeans
axis[2,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_unscaled']==4,'NObeyesdad'])
axis[2, 0].set_title('cluster 4')

# the categories that fall within group 5 for kmeans
axis[2,
1].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_unscaled']==5,'NObeyesdad'])
axis[2, 1].set_title('cluster 5')

# the categories that fall within group 6 for kmeans
```

```
axis[3,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_unscaled']==6,'NObeyesdad'])
axis[3, 0].set_title('cluster 6')

plt.tight_layout()
plt.show()

figure, axis = plt.subplots(4, 2)

# the categories that fall within group 0 for kmedioids
axis[0,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_unscaled']==0,'NObeyesdad'
])
axis[0, 0].set_title('cluster 0')

# the categories that fall within group 1 for kmedioids
axis[0,
1].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_unscaled']==1,'NObeyesdad'
])
axis[0, 1].set_title('cluster 1')

# the categories that fall within group 2 for kmedioids
axis[1,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_unscaled']==2,'NObeyesdad'
])
axis[1, 0].set_title('cluster 2')

# the categories that fall within group 3 for kmedioids
axis[1,
1].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_unscaled']==3,'NObeyesdad'
])
axis[1, 1].set_title('cluster 3')

# the categories that fall within group 4 for kmedioids
axis[2,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_unscaled']==4,'NObeyesdad'
])
axis[2, 0].set_title('cluster 4')

# the categories that fall within group 5 for kmedioids
axis[2,
1].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_unscaled']==5,'NObeyesdad'
])
axis[2, 1].set_title('cluster 5')
```

```
# the categories that fall within group 6 for kmedioids
axis[3,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_unscaled']==6,'NObeyesdad'
])
axis[3, 0].set_title('cluster 6')

plt.tight_layout()
plt.show()

# K-means clustering. 7 clusters for 7 different obesity levels
kmeans = KMeans(n_clusters=7, random_state=42)
kmeans_labels = kmeans.fit_predict(obesity_scaled)

# K-medioid clustering. 7 clusters for 7 different obesity levels
kmedoids = KMedoids(n_clusters=7, random_state=42)
kmedoids_labels = kmedoids.fit_predict(obesity_scaled)

# Combining the "predictions" from the kmeans and kmedioid approaches
cluster_map = pd.DataFrame()
cluster_map['data_index'] = data_encoded.index.values
cluster_map['cluster_kmeans_scaled'] = kmeans_labels
cluster_map['cluster_kmedioids_scaled'] = kmedoids_labels

#Combining the predictions with the origninal obestiy data
obesity_clustering = pd.concat([data_encoded, cluster_map], axis = 1)

#What is the prediction for kmeans clustering?
for i in set(obesity_clustering['NObeyesdad']):
    print(f"The kmeans scaled prediction for cluster {i} is group
        {mode(obesity_clustering.loc[obesity_clustering['cluster_kmeans_scaled']==i,'NObeyesd
        ad'])}")
    categories_in_cluster =
        list(obesity_clustering.loc[obesity_clustering['cluster_kmeans_scaled']==i,'NObeyesdad'])
    print(f"The purity of cluster {i} is
        {categories_in_cluster.count(mode(obesity_clustering.loc[obesity_clustering['cluster_km
        eans_scaled']==i,'NObeyesdad'])) / len(categories_in_cluster)}")

#What is the prediction for kmedioid clustering?
for i in set(obesity_clustering['NObeyesdad']):
    print(f"The k medioids scaled prediction for cluster {i} is group
        {mode(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_scaled']==i,'NObeye
        sdad'])}")
    categories_in_cluster =
```

```
        list(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_scaled']==i,'NObeyesda
        d'])
    print(f"the purity of cluster {i} is
        {categories_in_cluster.count(mode(obesity_clustering.loc[obesity_clustering['cluster_km
        edioids_scaled']==i,'NObeyesdad'])) / len(categories_in_cluster)}")

figure, axis = plt.subplots(4, 2)

# the categories that fall within group 0 for kmeans
axis[0,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_scaled']==0,'NObeyesdad'])
axis[0, 0].set_title('cluster 0')

# the categories that fall within group 1 for kmeans
axis[0,
1].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_scaled']==1,'NObeyesdad'])
axis[0, 1].set_title('cluster 1')

# the categories that fall within group 2 for kmeans
axis[1,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_scaled']==2,'NObeyesdad'])
axis[1, 0].set_title('cluster 2')

# the categories that fall within group 3 for kmeans
axis[1,
1].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_scaled']==3,'NObeyesdad'])
axis[1, 1].set_title('cluster 3')

# the categories that fall within group 4 for kmeans
axis[2,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_scaled']==4,'NObeyesdad'])
axis[2, 0].set_title('cluster 4')

# the categories that fall within group 5 for kmeans
axis[2,
1].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_scaled']==5,'NObeyesdad'])
axis[2, 1].set_title('cluster 5')

# the categories that fall within group 6 for kmeans
axis[3,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmeans_scaled']==6,'NObeyesdad'])
axis[3, 0].set_title('cluster 6')

plt.tight_layout()
```

```
plt.show()

figure, axis = plt.subplots(4, 2)

# the categories that fall within group 0 for kmedioids
axis[0,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_scaled']==0,'NObeyesdad'])
axis[0, 0].set_title('cluster 0')

# the categories that fall within group 1 for kmedioids
axis[0,
1].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_scaled']==1,'NObeyesdad'])
axis[0, 1].set_title('cluster 1')

# the categories that fall within group 2 for kmedioids
axis[1,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_scaled']==2,'NObeyesdad'])
axis[1, 0].set_title('cluster 2')

# the categories that fall within group 3 for kmedioids
axis[1,
1].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_scaled']==3,'NObeyesdad'])
axis[1, 1].set_title('cluster 3')

# the categories that fall within group 4 for kmedioids
axis[2,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_scaled']==4,'NObeyesdad'])
axis[2, 0].set_title('cluster 4')

# the categories that fall within group 5 for kmedioids
axis[2,
1].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_scaled']==5,'NObeyesdad'])
axis[2, 1].set_title('cluster 5')

# the categories that fall within group 6 for kmedioids
axis[3,
0].hist(obesity_clustering.loc[obesity_clustering['cluster_kmedioids_scaled']==6,'NObeyesdad'])
axis[3, 0].set_title('cluster 6')

plt.tight_layout()
plt.show()
```