Martinez assn 7

October 21, 2020

1 Martinez Assn 7

1.1 Unsupervised Learning - Kmeans Clustering

Access the "DSC-540 Data Sets" document found in the course materials, focusing on data sets best suited for unsupervised cluster detection.

Select one of the 90+ datasets as the data source for your project.

Familiarize yourself with the KMeans package in Python and its use in a Jupyter notebook by utilizing the Learn KMeans resource within the topic materials.

Referring to the chosen dataset, formulate three questions worth asking about the data, and explain why it is important and beneficial to ask these questions. Ask yourself: "What can I learn by grouping similar data points together and discovering underlying patterns?" While the readings for this topic are comprehensive, you may use the GCU digital library to find additional articles describing the use of kmeans.

Perform a Kmeans cluster analysis, plot the results, and interpret them.

Use the results above and answer the questions you previously formulated. For each question write the answer mathematical/quantitative terms. Explain the results and the meaning of the patterns you uncovered. Use additional plots to support your arguments.

[343]:
```
import pandas as pd # dataframe
import numpy as np # for array math
from matplotlib import pyplot as plt # plotting
from sklearn.cluster import KMeans # the algorithm
from sklearn.decomposition import PCA # Need this to reduce dimensionality
from sklearn.preprocessing import StandardScaler # Gotta make sure everything␣
↪is unit norm
from sklearn.preprocessing import OrdinalEncoder # To encode the variables.
from mpl_toolkits.mplot3d import Axes3D
import matplotlib
from sklearn.preprocessing import LabelEncoder
%matplotlib inline
```

1.2 The Dataset

I work for a large insurance company, specific in the preventive health space. My role is to analyze the data aand provide reports that will be useful to getting members to use benefits such as gym reimbursements, nutrition coaching, etc. Therefore, I chose a dataset that would largely reflect something I would see in the real world.

The Dataset is from the UCI Machine learning repository. It is the Estimation of obesity levels based on eating habits and physical condition Data Set. There is a mix of categorical and numerical values so some data cleanup will be required as KMeans doesn't allow for non-numeric data. Also because certain things such as gender may provide good information on variance, dichotomous variables don't lend themselves well to KMeans since the average of two values is not as meaningful (especially when they are categorical). PCA will be used to reduce the dimensionality prior to running the code.

The attribute breakdown is as follows: - Frequent consumption of high caloric food (FAVC), - Frequency of consumption of vegetables (FCVC), - Number of main meals (NCP), - Consumption of food between meals (CAEC), - Consumption of water daily (CH20), - Consumption of alcohol (CALC). - Calories consumption monitoring (SCC) - Physical activity frequency (FAF) - Time using technology devices (TUE) - Transportation used (MTRANS) - Gender, - Age, - Height - Weight

The main questions for the dataset will be:

- How does the consumption of water impact the eating habits of people in particular groups? In other words, does water have a direct relationship with a specific weight category?
- I presume that people who use public transport or biking will be more likely to fall into a cluster of lower weight people
- Are there drastic similiarities or differences when it comes to male/female differences in the data clusters?

The dataset can be found here: http://archive.ics.uci.edu/ml/datasets/Estimation+of+obesity+levels+based+on+

1.3 Preliminary Data Analysis

[151]:
```
# Load up the data first.
data = pd.read_csv('../ObesityDataSet_raw_and_data_sinthetic.csv')

# This data is labeled, so it may be worth it to remove that and see if it␣
↪changes things.
```

[340]:
```
# View the initial rows to ensure that our data loaded properly.
data.head()
```

[340]:
```
   Gender   Age  Height  Weight family_history_with_overweight FAVC  FCVC  \
0  Female  21.0    1.62    64.0                            yes   no   2.0
1  Female  21.0    1.52    56.0                            yes   no   3.0
2    Male  23.0    1.80    77.0                            yes   no   2.0
3    Male  27.0    1.80    87.0                             no   no   3.0
4    Male  22.0    1.78    89.8                             no   no   2.0

   NCP       CAEC SMOKE  CH2O  SCC  FAF  TUE        CALC  \
0  3.0  Sometimes    no   2.0   no  0.0  1.0          no
1  3.0  Sometimes   yes   3.0  yes  3.0  0.0   Sometimes
```

```
2  3.0  Sometimes    no   2.0   no  2.0  1.0  Frequently
3  3.0  Sometimes    no   2.0   no  2.0  0.0  Frequently
4  1.0  Sometimes    no   2.0   no  0.0  0.0   Sometimes

                  MTRANS           NObeyesdad
0  Public_Transportation        Normal_Weight
1  Public_Transportation        Normal_Weight
2  Public_Transportation        Normal_Weight
3                Walking   Overweight_Level_I
4  Public_Transportation  Overweight_Level_II
```

[341]:

```
# Check for any nulls just to be sure. I've not gone through this data so there␣
 ↪may be some we need to handle.

data[data.isnull().any(axis=1)]
```

[341]:

```
Empty DataFrame
Columns: [Gender, Age, Height, Weight, family_history_with_overweight, FAVC,
FCVC, NCP, CAEC, SMOKE, CH2O, SCC, FAF, TUE, CALC, MTRANS, NObeyesdad]
Index: []
```

[342]:

```
# Look at the distributions of the numeric fields and see if there is anything␣
 ↪striking

data.describe()
```

[342]:

```
               Age       Height       Weight         FCVC          NCP  \
count  2111.000000  2111.000000  2111.000000  2111.000000  2111.000000
mean     24.312600     1.701677    86.586058     2.419043     2.685628
std       6.345968     0.093305    26.191172     0.533927     0.778039
min      14.000000     1.450000    39.000000     1.000000     1.000000
25%      19.947192     1.630000    65.473343     2.000000     2.658738
50%      22.777890     1.700499    83.000000     2.385502     3.000000
75%      26.000000     1.768464   107.430682     3.000000     3.000000
max      61.000000     1.980000   173.000000     3.000000     4.000000

              CH2O          FAF          TUE
count  2111.000000  2111.000000  2111.000000
mean      2.008011     1.010298     0.657866
std       0.612953     0.850592     0.608927
min       1.000000     0.000000     0.000000
25%       1.584812     0.124505     0.000000
50%       2.000000     1.000000     0.625350
75%       2.477420     1.666678     1.000000
max       3.000000     3.000000     2.000000
```

1.4 Preprocessing

The data contains 17 attributes. There are some factors and some numeric. So in order to deal with this, I will encode the categorical variables.

It should be noted that this is not really the best way to handle categorical variables with K-Means. By assigning arbitrary numbers to represent values, we implicitly provide the algorithm misleading data that it has no way to interpret (Gopal, 2020, p. 375). This can lead to improper results. That being said, the number of categorical values in those variables is low so the impact will be less than, say, encoding state.

[345]:
```
# create a list of the variables which are categorical
cat_vars = ['Gender', 'family_history_with_overweight', 'FAVC', 'CAEC',␣
 ↪'SMOKE', 'SCC', 'CALC', 'MTRANS', 'NObeyesdad']

# create an instance of the label encoder class. This is because the fields are␣
 ↪not ordinal, and we are doing them
# one at a time meaning that each array will be 1D.
le = LabelEncoder()

# apply the encoder
l_data = data[cat_vars].apply(le.fit_transform)

# check to make sure it worked
l_data.head()
```

[345]:
```
   Gender  family_history_with_overweight  FAVC  CAEC  SMOKE  SCC  CALC  \
0       0                               1     0     2      0    0     3
1       0                               1     0     2      1    1     2
2       1                               1     0     2      0    0     1
3       1                               0     0     2      0    0     1
4       1                               0     0     2      0    0     2

   MTRANS  NObeyesdad
0       3           1
1       3           1
2       3           1
3       4           5
4       3           6
```

[346]:
```
# Now join the encoded data back to the original numeric values

y_data = l_data.join(data._get_numeric_data())

# double check that everything is numeric
y_data.dtypes
```

[346]:
```
Gender                            int32
family_history_with_overweight    int32
FAVC                              int32
CAEC                              int32
SMOKE                             int32
SCC                               int32
CALC                              int32
MTRANS                            int32
NObeyesdad                        int32
Age                             float64
Height                          float64
Weight                          float64
FCVC                            float64
NCP                             float64
CH2O                            float64
FAF                             float64
TUE                             float64
dtype: object
```

1.5 Scaling and Dimension Reduction

Because K-Means uses the Euclidean distance, it's critical that the values are scaled to prevent any issues creating a massive confusion among the centroids. While this is fine to do for this example, it would be more useful in the future to use something such as K-Modes which is much better at handling encoded categorical variables for clustering (He, 2006).

K-means can also have dimensionality issues, so it's useful to reduce the dimensionality a bit. These issues arise from the multiple distances that can be of varying impact. This is a relatively low dimension dataset, but for the benefits of later visualization, I'll go ahead and reduce the variables down to two.

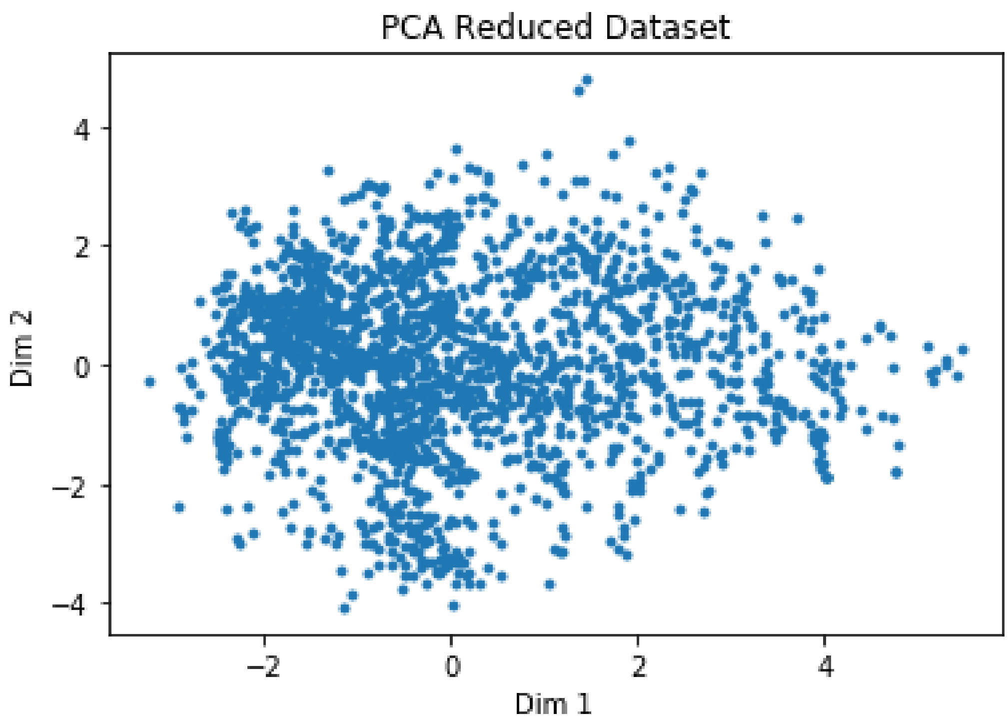
[347]:
```
# Create an instance of the standard scaler and apply it to the all-numeric␣
↪dataframe

sc = StandardScaler()
n_data = sc.fit_transform(y_data)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:645:
DataConversionWarning: Data with input dtype int32, float64 were all converted
to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:464:
DataConversionWarning: Data with input dtype int32, float64 were all converted
to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
```

[348]:
```
# Build a PCA instance with just two components.
# Fit the pca to the dataset and apply the transformations.
```

```
pca = PCA(n_components=2)
f_data = pca.fit_transform(n_data)
```

[280]:
```
# Plot our newly formed dimensions
plt.plot(f_data[:, 0], f_data[:,1], '.')
plt.title('PCA Reduced Dataset')
plt.xlabel('Dim 1')
plt.ylabel('Dim 2')
plt.show()
```



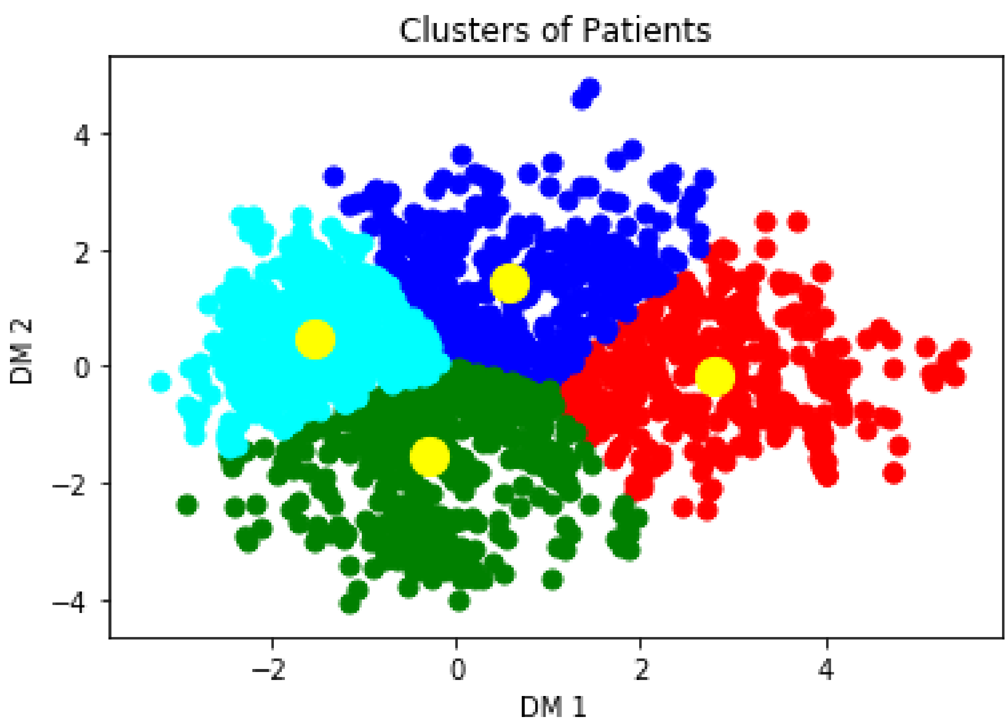
There don't seem to be any real clear clusters, but there is some concentration on the left-hand side. However, variance is clearly different on the right hand side points which may result in some issues. The density of the clusters impacts k-means. The default function parameter of starting point uses the K-means++ function, which ensures that the starting centroids are sufficiently distant from each other to prevent converging at a local minimum (Sharma, 2019).

1.6 Build a K-means model

[349]:
```
# Arbitrary starting point. Chosen because while there are six weight labels,␣
↪there are only four distinct ones.
k = 4

kmeans = KMeans(k).fit(f_data)
```

```
y_preds = kmeans.fit_predict(f_data)
```

[282]:
```
plt.scatter(f_data[y_preds==0, 0], f_data[y_preds==0, 1], s=50, c='red', label␣
↪='Cluster 1')
plt.scatter(f_data[y_preds==1, 0], f_data[y_preds==1, 1], s=50, c='blue', label␣
↪='Cluster 2')
plt.scatter(f_data[y_preds==2, 0], f_data[y_preds==2, 1], s=50, c='green',␣
↪label ='Cluster 3')
plt.scatter(f_data[y_preds==3, 0], f_data[y_preds==3, 1], s=50, c='cyan', label␣
↪='Cluster 4')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],␣
↪s=200, c='yellow', label = 'Centroids')
plt.title('Clusters of Patients')
plt.xlabel('DM 1')
plt.ylabel('DM 2')
plt.show()
```



[351]:
```
# View the cluster centers

kmeans.cluster_centers_
```

[351]: array([[-1.51897069, 0.49933454],
        [2.79124359, -0.17649972],
        [0.57249624, 1.43922218],
        [-0.29142815, -1.51883221]])

1.7 The return to normalcy

Principal components are not super useful to stakeholders since it is difficult to derive meaning from them. However, reconstruction of the data points is possible. It works in a similar way to SVD or certain types of image processing: remove some non essential information from the object, then reconstruct a lower resolution object with the data points that explain most of the variance (amoeba, 2016).

$$PCAreconstruction = PCscores\ Eigenvectors\ + Mean$$

[352]:
```
# Get the mean of the original dataset

mu = np.mean(n_data, axis=0)
```

[353]:
```
nComp = 2
Xhat = np.dot(pca.transform(n_data)[:,:nComp], pca.components_[:nComp,:])
Xhat += mu
```

[354]:
```
Xhat[0,]
```

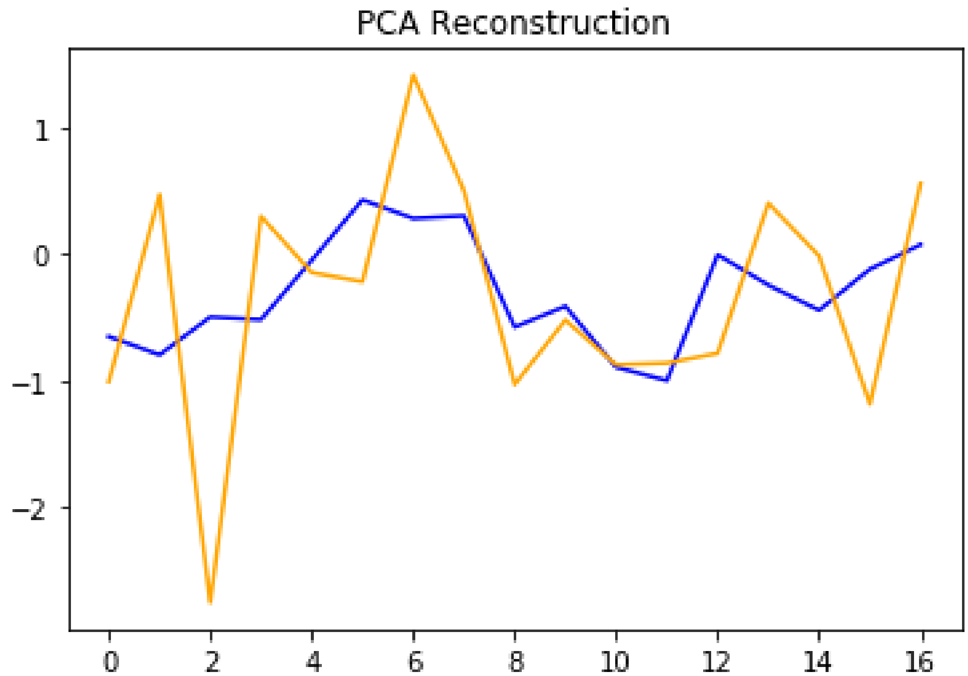
[354]:
```
array([-0.65224806, -0.79732732, -0.49695708, -0.5180365 , -0.04848812,
        0.4304274 ,  0.28328097,  0.30400024, -0.57715783, -0.41189932,
       -0.89542006, -1.00173199, -0.00464169, -0.24449349, -0.44365243,
       -0.11916094,  0.07538557])
```

[355]:
```
n_data[0,]
```

[355]:
```
array([-1.01191369,  0.47229133, -2.75976929,  0.30034556, -0.14590027,
       -0.21827203,  1.4191716 ,  0.50333674, -1.03279553, -0.52212439,
       -0.87558934, -0.86255819, -0.7850187 ,  0.40415272, -0.01307326,
       -1.18803911,  0.56199675])
```

[359]:
```
# A plot here shows the differences between the data points
# The massive drop is likely due to the fact that reducing 17 columns down to
 ↪two lost too much variance
# although further analysis would be required to be sure.
plt.plot(Xhat[0,], color="blue")
plt.plot(n_data[0,], color="orange")
plt.title('PCA Reconstruction')
ax.legend()
```

[359]: <matplotlib.legend.Legend at 0x1e3e6cbdfd0>



[260]:
```python
# Now that the points have been assigned clusters, we can join back to the
↪original data to do some analysis on
# the clustered sets.

final = pd.DataFrame(y_preds)
final = final.join(data)
final.head(15)
```

[260]:
```
    0  Gender   Age  Height  Weight family_history_with_overweight FAVC  FCVC  \
0   0  Female  21.0    1.62    64.0                            yes   no   2.0
1   0  Female  21.0    1.52    56.0                            yes   no   3.0
2   1    Male  23.0    1.80    77.0                            yes   no   2.0
3   1    Male  27.0    1.80    87.0                             no   no   3.0
4   2    Male  22.0    1.78    89.8                             no   no   2.0
5   2    Male  29.0    1.62    53.0                             no  yes   2.0
6   2  Female  23.0    1.50    55.0                            yes  yes   3.0
7   1    Male  22.0    1.64    53.0                             no   no   2.0
8   1    Male  24.0    1.78    64.0                            yes  yes   3.0
9   1    Male  22.0    1.72    68.0                            yes  yes   2.0
10  1    Male  26.0    1.85   105.0                            yes  yes   3.0
11  1  Female  21.0    1.72    80.0                            yes  yes   2.0
12  1    Male  22.0    1.65    56.0                             no   no   3.0
```

```
13  3    Male  41.0    1.80    99.0                                  no  yes   2.0
14  1    Male  23.0    1.77    60.0                                 yes  yes   3.0

    NCP        CAEC SMOKE  CH2O  SCC  FAF  TUE        CALC  \
0   3.0   Sometimes    no   2.0   no  0.0  1.0          no
1   3.0   Sometimes   yes   3.0  yes  3.0  0.0   Sometimes
2   3.0   Sometimes    no   2.0   no  2.0  1.0  Frequently
3   3.0   Sometimes    no   2.0   no  2.0  0.0  Frequently
4   1.0   Sometimes    no   2.0   no  0.0  0.0   Sometimes
5   3.0   Sometimes    no   2.0   no  0.0  0.0   Sometimes
6   3.0   Sometimes    no   2.0   no  1.0  0.0   Sometimes
7   3.0   Sometimes    no   2.0   no  3.0  0.0   Sometimes
8   3.0   Sometimes    no   2.0   no  1.0  1.0  Frequently
9   3.0   Sometimes    no   2.0   no  1.0  1.0          no
10  3.0  Frequently    no   3.0   no  2.0  2.0   Sometimes
11  3.0  Frequently    no   2.0  yes  2.0  1.0   Sometimes
12  3.0   Sometimes    no   3.0   no  2.0  0.0   Sometimes
13  3.0   Sometimes    no   2.0   no  2.0  1.0  Frequently
14  1.0   Sometimes    no   1.0   no  1.0  1.0   Sometimes

                   MTRANS           NObeyesdad
0   Public_Transportation        Normal_Weight
1   Public_Transportation        Normal_Weight
2   Public_Transportation        Normal_Weight
3                 Walking   Overweight_Level_I
4   Public_Transportation  Overweight_Level_II
5              Automobile        Normal_Weight
6               Motorbike        Normal_Weight
7   Public_Transportation        Normal_Weight
8   Public_Transportation        Normal_Weight
9   Public_Transportation        Normal_Weight
10  Public_Transportation       Obesity_Type_I
11  Public_Transportation  Overweight_Level_II
12  Public_Transportation        Normal_Weight
13             Automobile       Obesity_Type_I
14  Public_Transportation        Normal_Weight
```

1.8 Post-model analysis

Now that we've got the dataset recombobulated with the labels, we can begin to answer the initial questions. To recap:

- How does the consumption of water impact the eating habits of people in particular groups? In other words, does water have a direct relationship with a specific weight category?
- I presume that people who use public transport or biking will be more likely to fall into a cluster of lower weight people
- Are there drastic similiarities or differences when it comes to male/female differences in the data clusters?

[360]:
```
# Separate the clusters out into individual dataframes for convenience
# ... and to waste memory

gr0 = final[final[0] == 0]
gr1 = final[final[0] == 1]
gr2 = final[final[0] == 2]
gr3 = final[final[0] == 3]
```

[330]:
```
# It appears that the mean weight of the clusters was a large deciding factor.

print(gr0['Weight'].mean())
print(gr1['Weight'].mean())
print(gr2['Weight'].mean())
print(gr3['Weight'].mean())
```

```
54.829932158192115
72.24884708482138
89.91119202302635
108.90140438944363
```

[363]:
```
# It's particularly interesting that the dataset grouped mostly women in the␣
↪smallest cluster and metn in the largest.
# Weight alone is likely not sufficient to determine cluster attributes. Let's␣
↪dig into it a bit more.

print(gr0['Gender'].value_counts())
print(gr1['Gender'].value_counts())
print(gr2['Gender'].value_counts())
print(gr3['Gender'].value_counts())
```

```
Female    335
Male       19
Name: Gender, dtype: int64
Male      344
Female    104
Name: Gender, dtype: int64
Female    458
Male      150
Name: Gender, dtype: int64
Male      555
Female    146
Name: Gender, dtype: int64
```

[367]:
```
gr0[gr0['Gender'] == 'Male']

print(gr0['NObeyesdad'].value_counts())
```

```
# Interestingly this one person was in the lowest cluster but was overweight.
# It's clear that the clusters used the other values too.
print(gr0[gr0['NObeyesdad'] == 'Obesity_Type_II'])
```

```
Insufficient_Weight    141
Normal_Weight          131
Overweight_Level_I      42
Overweight_Level_II     28
Obesity_Type_I          11
Obesity_Type_II          1
Name: NObeyesdad, dtype: int64
    0  Gender   Age  Height  Weight family_history_with_overweight FAVC  FCVC  \
90  0  Female  25.0    1.63    93.0                             no   no   3.0

    NCP    CAEC SMOKE  CH2O SCC  FAF  TUE CALC                 MTRANS  \
90  4.0  Always    no   1.0  no  2.0  0.0   no  Public_Transportation

         NObeyesdad
90  Obesity_Type_II
```

1.8.1 Question 1

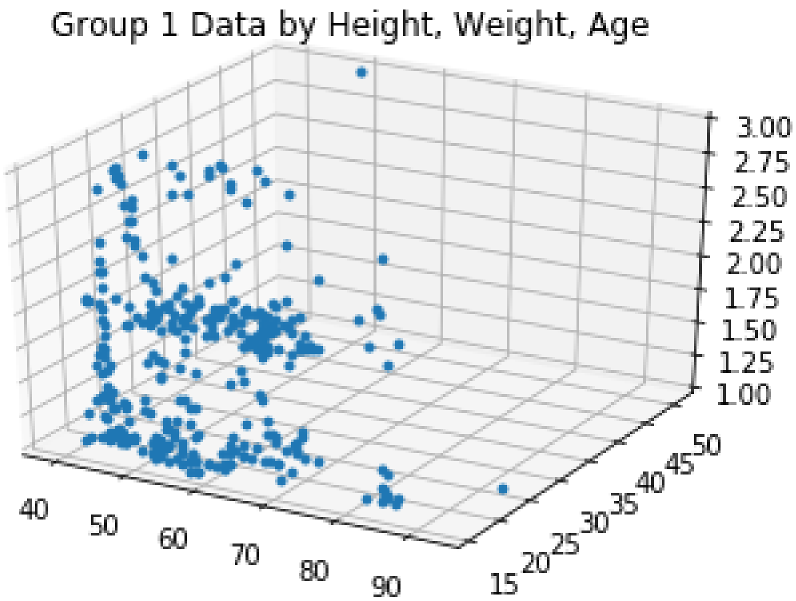
How does the consumption of water impact the eating habits of people in particular groups? In other words, does water have a direct relationship with a specific weight category?

[411]:
```
# Looking at the mean values for the water consumption, it does not appear that␣
↪there are any
# noticeable differences in the clusters and water consumption.
# Deeper analysis would be required to sort out differences between
# men and women within the groups.

print(gr0['CH2O'].mean())
print(gr1['CH2O'].mean())
print(gr2['CH2O'].mean())
print(gr3['CH2O'].mean())
```

```
1.6475033135593218
2.120320154017858
1.928378226973683
2.1873587874465055
```

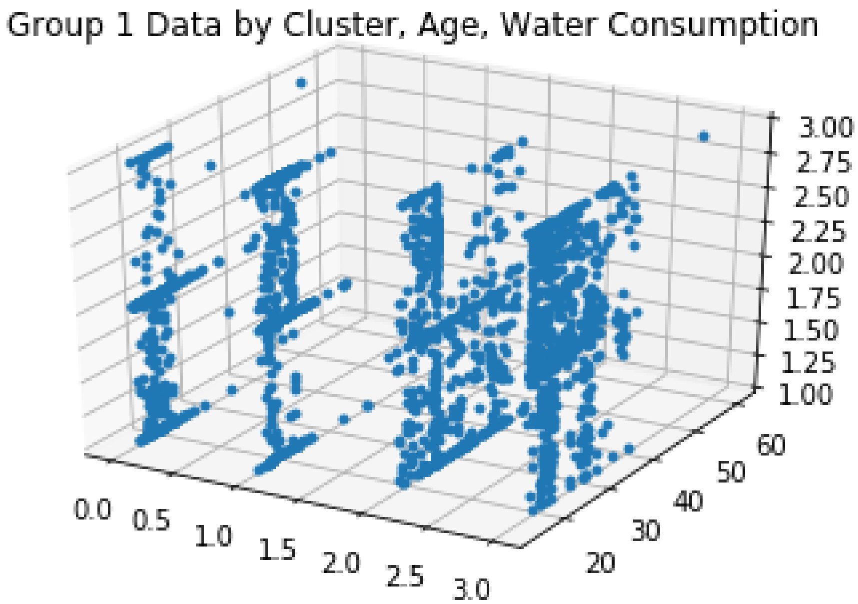
[418]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
plt.plot(gr0['Weight'], gr0['Age'], gr0['CH2O'], '.')
plt.title("Group 1 Data by Height, Weight, Age")
```

[418]: Text(0.5, 0.92, 'Group 1 Data by Height, Weight, Age')



[426]:
```
# It's clear here that group 4, the heaviest, drinks the most water.
# It could be a direct relationship with meals consumed per day
# More analysis would be needed.

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
plt.plot(final[0], final['Age'], final['CH2O'], '.')
plt.title("Group 1 Data by Cluster, Age, Water Consumption")
```

[426]:
```
Text(0.5, 0.92, 'Group 1 Data by Cluster, Age, Water Consumption')
```



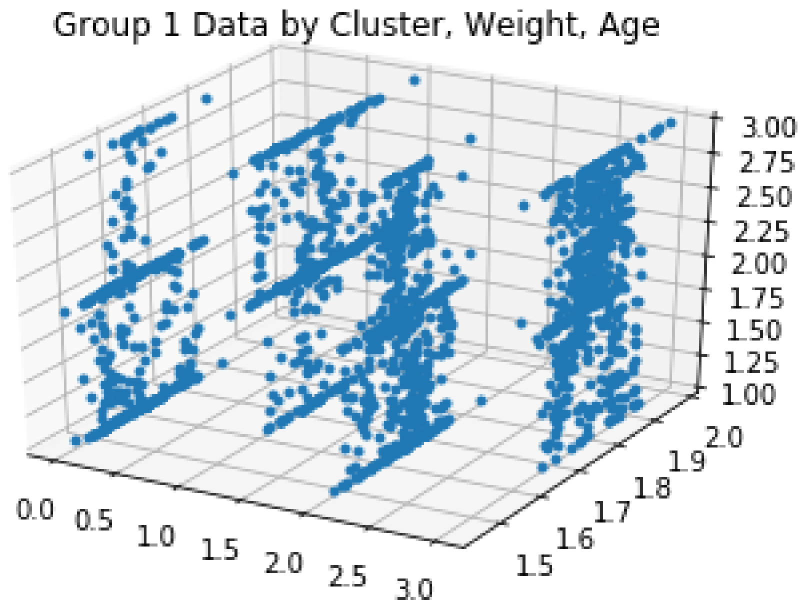
1.8.2 Question 2

I presume that people who use public transport or biking will be more likely to fall into a cluster of lower weight people.

The below data clearly show that walking and biking are reduced in the heavier clusters. The initial assumption is that those groups may simply be less active. However, it is difficult to come to that conclusion without controlling for socioeconomic status (which we don't have) or the oversampling of particular genders in each cluster. It may be that some people do not live in places where biking is feasible.

[425]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
plt.plot(final[0], final['Height'], final['CH2O'], '.')
plt.title("Group 1 Data by Cluster, Weight, Age")
```

[425]:
```
Text(0.5, 0.92, 'Group 1 Data by Cluster, Weight, Age')
```



[444]:
```
gr0.groupby(['MTRANS', 'Gender'])[['NObeyesdad']].count()
```

[444]:

		NObeyesdad
MTRANS	Gender	
Automobile	Female	18
	Male	1
Motorbike	Male	2
Public_Transportation	Female	300
	Male	15
Walking	Female	17
	Male	1

[443]:
```
gr1.groupby(['MTRANS', 'Gender'])[['NObeyesdad']].count()
```

[443]:

		NObeyesdad
MTRANS	Gender	
Automobile	Female	2
	Male	62
Bike	Male	6
Motorbike	Male	6
Public_Transportation	Female	98
	Male	245
Walking	Female	4
	Male	25

[441]:
```
gr2.groupby(['MTRANS', 'Gender'])[['NObeyesdad']].count()
```

[441]:
```
                                 NObeyesdad
MTRANS                 Gender
Automobile             Female           146
                       Male              60
Bike                   Male               1
Motorbike              Female             2
                       Male               1
Public_Transportation  Female           310
                       Male              88
```

[442]:
```
gr3.groupby(['MTRANS', 'Gender'])[['NObeyesdad']].count()
```

[442]:
```
                                 NObeyesdad
MTRANS                 Gender
Automobile             Male             168
Public_Transportation  Female           146
                       Male             378
Walking                Male               9
```

1.8.3 Question 3

Are there drastic similiarities or differences when it comes to male/female differences in the data clusters?

This was answered above, particulary by the stark gender offsets in each group. Women dominate the lower numbers, while men are overrepresented in the upper ones. Further analysis would be useful to determine what traits each gender has that differ across the groups. The vast majority of all people did not smoke, so this was statistically insignificant when determining the clustering.

[446]:
```
gr0.groupby(['NObeyesdad','Gender'])[['NObeyesdad']].count()
```

[446]:
```
                             NObeyesdad
NObeyesdad          Gender
Insufficient_Weight Female          141
Normal_Weight       Female          114
                    Male             17
Obesity_Type_I      Female           11
Obesity_Type_II     Female            1
Overweight_Level_I  Female           41
                    Male              1
Overweight_Level_II Female           27
                    Male              1
```

[447]:
```
gr1.groupby(['NObeyesdad','Gender'])[['NObeyesdad']].count()
```

[447]:
```
                             NObeyesdad
NObeyesdad          Gender
Insufficient_Weight Female           32
```

```
                     Male            93
Normal_Weight        Female          14
                     Male           125
Obesity_Type_I       Female          38
                     Male            70
Obesity_Type_II      Male             3
Overweight_Level_I   Female          12
                     Male            24
Overweight_Level_II  Female           8
                     Male            29
```

[448]:
```
gr2.groupby(['NObeyesdad','Gender'])[['NObeyesdad']].count()
```

[448]:
```
                             NObeyesdad
NObeyesdad           Gender
Normal_Weight        Female          13
                     Male             4
Obesity_Type_I       Female         107
                     Male            16
Obesity_Type_II      Female           1
                     Male            66
Obesity_Type_III     Female         189
Overweight_Level_I   Female          83
                     Male            26
Overweight_Level_II  Female          65
                     Male            38
```

[451]:
```
gr3.groupby(['NObeyesdad','Gender'])[['NObeyesdad']].count()
```

[451]:
```
                             NObeyesdad
NObeyesdad           Gender
Insufficient_Weight  Male             6
Obesity_Type_I       Male           109
Obesity_Type_II      Male           226
Obesity_Type_III     Female         134
                     Male             1
Overweight_Level_I   Female           9
                     Male            94
Overweight_Level_II  Female           3
                     Male           119
```

[453]:
```
gr0.groupby(['Gender', 'SMOKE'])[['NObeyesdad']].count()
```

[453]:
```
              NObeyesdad
Gender SMOKE
Female no            331
       yes             4
```

```
Male   no             19
```

[454]:
```
gr1.groupby(['Gender', 'SMOKE'])[['NObeyesdad']].count()
```

[454]:
```
              NObeyesdad
Gender SMOKE
Female no            100
       yes             4
Male   no            334
       yes            10
```

[455]:
```
gr2.groupby(['Gender', 'SMOKE'])[['NObeyesdad']].count()
```

[455]:
```
              NObeyesdad
Gender SMOKE
Female no            451
       yes             7
Male   no            148
       yes             2
```

[456]:
```
gr3.groupby(['Gender', 'SMOKE'])[['NObeyesdad']].count()
```

[456]:
```
              NObeyesdad
Gender SMOKE
Female no            146
Male   no            538
       yes            17
```

1.9 References

amoeba (https://stats.stackexchange.com/users/28666/amoeba), How to reverse PCA and reconstruct original variables from several principal components?, URL (version: 2017-04-13): https://stats.stackexchange.com/q/229093

Gopal, M. (2020). Applied Data Science. McGraw-Hill, New York. https://viewer.gcu.edu/MTyXCd

He, Z. (2006). Approxmation Algorithms for K-Modes Clustering. Harbin Institute of Technology. From Arxiv. https://arxiv.org/abs/cs/0603120

Sharma, P. (2019). The Most Comprehensive Guide to K-Means Clustering You'll Ever Need. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

[ ]:
```

```