Obesity Level Analysis of Adult Population in Latin America

Cody Le
Kokila Maddi
DePaul University
DSC 478
Fall 2021

November 20, 2021

Executive Summary

In the last three decades, a dramatic increase and rise in obesity rates has occurred in Latin America. The region has declared obesity as a public health concern which has been exacerbated by changes in economy, shift in workforce, and changes in diet. Two key factors have been determined as the cause in the rise in obesity: increase in consuming saturated fats and shift to a sedentary lifestyle. This analysis explores obesity levels among adults from three regions of Latin America to determine which specific eating habits or daily activities most affect the classification of obesity levels. Naturally, the expectation is that eating habits such as consuming high calorie foods and eating between meals along with decrease in daily physical activity would be the most important factors in determining obesity levels. Other factors such as gender and age are also compared since physical differences exist by gender and ability differences exist by age.

The dataset contains a total of 17 attributes and 2,111 instances. The class label is obesity level which is determined by body mass index (BMI). The index is considered a nutritional status index developed as a risk indicator for risk with the higher the BMI the higher chance of disease including premature death. Most of the features are categorical variables which include six eating habit attributes and five daily activities attributes. The remaining variables are gender, age, weight, height, and family history of obesity which are considered biological attributes. The data is transformed and normalized for the analysis and dummy variables are used for categorical features. To explore how age groups play a role in classifying obesity levels, the age attribute is binned by generational groups representing three age groups: Generation-Z, Millennials, and Generation-X and Boomers. These age groups will be used for cluster analysis exploration and feature selection.

The methodologies used for this analysis include data exploration and feature transformations which are part of preprocessing, cluster analysis exploration, classification and model selection, and feature selection. K-means algorithm is used for cluster analysis and performed on the original dataset as well as the transformed dataset with age groups to determine if patterns exist in the data. Classification models are built using a pipeline which bundles preprocessing and classifier models and returns a classification report for various models. The best classifier model is used in feature selection which compares the top 15% of the most important features in classifying obesity levels between the original dataset and each age group dataset. The top features will be evaluated in determining which factors are most salient in determining obesity levels.

The cluster analysis exploration revealed that a pattern exists between male and female gender. This aligns with the original prediction that gender differences play a role in classifying obesity levels. The analysis did not reveal any significant pattern in the generational age groups. Decision Tree was determined as the best classifier model with an accuracy of around 94%. The classifier model had the best accuracy and performance using the original dataset. The accuracy decreased steadily when the model was performed on age group datasets. The results of the feature selection using the classifier model revealed that the most important factors affecting the classification of obesity levels are age, gender, weight, and family history with obesity. The original prediction stated that eating high calorie foods frequently, eating between meals, and having lower days of physical activity would be the most salient attributes in determining obesity levels. However, the results show that always eating vegetables with meals and frequently eating between meals are the most important factors besides gender, weight, and family history with obesity. Physical activity attributes were not the most important factors in classifying obesity levels. Although age is an important factor, an individual's generational group does not play a salient factor in classifying obesity level.

In summary, this analysis shows that obesity level as defined here is a nutritional status index as the classes are created using BMI. In analyzing the data set of the adult population in Mexico, Peru, and Columbia, we found that eating habit attributes are among the most important factors in classifying obesity level, specifically always eating vegetables with meals frequently and eating between meals frequently. Factors that are more salient than eating habits in determining the obesity level include gender, age, weight, height, and no family history with obesity. Daily physical activities are not among the most important factors in classifying obesity levels.

Contents and Contributions

Introduction

The World Health Organization (WHO) states that obesity has tripled since 1975. The WHO states that in 2016, more than 1.9 billion adults aged 18 and older were overweight and of these 650 million were obese. Being overweight is defined as abnormal or excessive fat accumulation that may increase the risk for noncommunicable diseases such as heart disease and stroke which was the leading cause of death in 2012, diabetes, osteoarthritis, and cancer including endometrial, breast, ovarian, prostate, liver, gallbladder, kidney, and colon cancer. Today, a simple Body Mass Index (BMI) which is a radio of weight-for-height is used to classify overweight and obesity in adults.

In the last few decades, a dramatic increase in obesity rates has occurred in Latin America, becoming a public health concern for the region mostly exasperated by regions undergoing industrialization resulting to lifestyle changes (Kain et. Al 2003). Statistics from the WHO show that obesity rates have increased every decade in both men and women of all age groups, but in the last few years the upward trend is particularity dramatic in men. Two key factors play a role in the upward trend in Latina America, dietary changes, and physical inactivity due to a shift in the workforce. With improved economies in the region, increases in income results to an increase energy consumption and ultimately an increase in consuming saturated fats. Shifts in the labor force in the more developed countries in Latin America from a labor workforce to a service-oriented workforce has led to a sedentary lifestyle with activities that involve less physical movement such as watching television, playing computer games, and increased use of motor vehicles (Kain et. Al 2003). Research in dietary consumption and sedentary lifestyle in Latin America has not yet been fully studied and more data is needed to further understand the reasons for trends in obesity.

The WHO believes that obesity is preventable. What causes obesity? One main cause of obesity is energy imbalance between calories consumed and calories expanded. Changes in dietary and physical activity patterns are direct results in recent years of changes in environment and societal demands. Moreover, malnutrition is another threat that low and middle-income countries experience which exacerbates the dietary imbalance. What specific dietary and physical activity patterns leads to obesity levels in adults? This analysis will explore recent data of eating habits and daily physical activity among adults from Latin America to determine which specific factors lead to an individual's obesity level. Are there other factors that influence an individual's obesity level such as gender and age which past research has shown may influence the upward trend in obesity? This analysis will examine this question and allow for a better understanding of what aspects of an individual's daily life can they focus on to maintain or change their obesity level.

Objective

This analysis seeks to analyze a dataset containing obesity levels among adults from Mexico, Peru, and Colombia to determine which specific eating habit or daily activities most affect the classification of obesity levels. It is predicted that eating high calorie foods frequently, eating between meals, and having lower days of physical activity are the most salient attributes to determining obesity levels. In addition, studies have shown that gender and age may play a role in rising obesity trends as such results should show that gender and age will affect the top attributes differently. It is predicted that eating habit attributes will be most salient among younger age groups and physical activity attributes will be most salient among elderly age group.

Data Schema and Preparation

The dataset was obtained from the University of California Irvine (UCI) Machine Learning Repository. 23% of the data was collected directly from users through an online survey platform and 77% of the data was simulated data. The dataset contains a total of 17 attributes and 2,111 instances. The original file size is 264KB in comma separated values format. The class label is NObeyesdad (Obesity Level) representing the obesity levels. The total number of features is 16, which corresponds to the 16 remaining attributes. Most of the attributes are categorical variables with only three attributes as numeric variables: Age, Height, and Weight. These three numeric variables were used to create the class level (which was pre-calculated from the dataset using BMI index). Some categorical attributes were already pre-processed and displayed using numeric numbers that correspond to categories instead of actual categories. All variables that were meant to be categorical that read as numeric were transformed back to categorical. Two variables FCVC (eating high calorie foods frequently) and CAEC (eating food between meals) were transformed to ordinal variables due to the nature of the response having an orderly occurrence. The table below shows the attributes, their original data type, category description (if applicable), and the transformed data type (if applicable) used for the analysis:

#	Attribute Name	Original Data Type	Categories	Final Data Type
1.	Gender	Categorical / Object	Male Female	Categorical / Object
2.	Age in Years	Numeric / Float		Numeric / Integer
3.	Height in Meters	Numeric / Float		Numeric / Float
4.	Weight in Kilograms	Numeric / Float		Numeric / Float
5.	family_history_with_overweight	Categorical / Object		Categorical / Object
6.	FAVC (eating high calorie foods frequently)	Categorical / Object	Yes No	Categorical / Object
7.	FCVC (eating vegetables in meals)	Numeric / Float	1 – Never 2 – Sometimes 3 – Always	Ordinal / Object
8.	NCP (number of main meals daily)	Numeric / Float	1 – ‘1’ 2 – ‘2’ 3 – ‘3’ 4 – ‘3+’	Categorical / Object
9.	CAEC (eating food between meals)	Categorical / Object	No Sometimes Frequently Always	Ordinal / Object
10.	Smoke	Categorical / Object	Yes No	Categorical / Object
11.	CH2O (water intake per day in liters)	Numeric / Float	1 – ‘Less than a liter’ 2 – ‘Between 1 and 2 L’	Categorical / Object

			3 – ‘More than 2 L’	
12.	SCC (monitor calories on a daily basis)	Categorical / Object	Yes No	Categorical / Object
13.	FAF (physical activity in number of days)	Numeric / Float	0 – ‘I do not have’ 1 – ‘1 or 2 days’ 2 – ‘2 or 4 days’ 3 – ‘4 or 5 days’	Categorical / Object
14.	TUE (time spent on technology)	Numeric / Float	0 – ‘0 – 2 hours’ 1 – ‘3 – 5 hours’ 2 – ‘More than 5 hours’	Categorical / Object
15.	CALC (alcohol intake)	Categorical / Object	I do not drink Sometimes Frequently Always	Categorical / Object
16.	MTRANS (means of transportation)	Categorical / Object	Automobile Motor Bike Bike Public-Transportation Walking	Categorical / Object
17.	NObeyesdad (accordining to BMI) ***--- Class Label ---***	Categorical / Object	Insufficient_Weight Normal_Weight Overweight_Level_I Overweight_Level_II Obesity_Type_I Obesity_Type_II Obesity_Type_III	Categorical / Object

The class label NObeyesdad will be used as the target variable which consist of seven classes: Insufficient_Weight corresponds to a Body Mass Index (BMI) of less than 18.5, Normal_Weight corresponds to a body mass index of 18.5 to 24.9, Overweight_Level_I corresponds to a body mass index of 25 to 26.9, Overweight_Level_II corresponds to a body mass index of 27.0 to 29.9, Obesity_Type_I corresponds to a body mass index of 30 to 34.9, Obesity_Type_II corresponds to a body mass index of 35.0 to 39.9, and Obesity_Type_III corresponds to a body mass index of over 40. The classes were calculated using the BMI which is weight in kilograms divided by height squared. The WHO states that BMI ranges are based on the effect of excessive body fat and risk of disease or death. The index is developed as a risk indicator of disease. The higher the BMI, the higher risk of diseases including premature death, cardiovascular diseases, high blood pressure, osteoarthritis, some cancers, and diabetes. The index and the classes associated at each level of the index are considered nutritional statuses. The ideal BMI is normal weight, which is corresponds to 18.5 to 24.9, any range below this range or above this range are considered higher risk for disease.

The class label will be transformed to numeric for the analysis with 0 - representing Insufficient_Weight, 1 - representing Normal_Weight, 2 - representing Overweight_Level_I, 3 - representing Overweight_Level_II, 4 - representing Obesity_Type_I, 5 – representing Obesity_Type_II, and 6 – representing Obesity_Type_III. All categorical and ordinal features will be transformed to dummy variables for the analysis. The dummy variables will be used to represent the numeric version of the feature. With the dummy variables, the total feature size for the analysis is 43.

Additional datasets were created for the exploration of the data through clustering analysis. The age attribute was transformed to categorical variable based on age group generation. Age group generations were selected because of the common trends and traits associated with different generational groups. The age groups were transformed by binning the ages based on the following generations from the Beresford Research group which defined the generations based on U.S. Census Bureau and data from Pew Research Center: Gen-Z includes ages 9 – 24, Millennials includes ages 25 – 40, Gen-X includes ages 41 – 56, and Boomers include ages 57 to 66. For this analysis, Gen-X and Boomer were combined into one age group due to the low instances in the data for this category. The three age groups are separated into three separate datasets. The Gen-Z age group contains 1,353 instances, the Millennials age group contains 717 instances, and the Gen-X and Boomers age group contains 41 instances. In addition to analyzing the full dataset, classification will also be performed on each age group dataset to determine most important features and compare results for evaluation.

Approach

The approach and methodology in this analysis first begins with data preprocessing which includes data exploration and clustering analysis. The data exploration provides a visual of key features being explored such as gender as well as the class label being explored which is obesity level by BMI. The clustering analysis is performed as an exploratory analysis to determine if there are any patterns in the full dataset and if any patterns exist when the age groups are binned to the generational groups. The clustering analysis uses K-means clustering algorithm which is a method of vector quantization with the goal of partitioning specified number of observations into specified number of clusters based on the nearest mean. Next, the data analysis is performed by performing classification and model selection on the data. Several classifier models are fit to the dataset including K-Nearest Neighbor, Decision Tree, Stochastic Gradient Descent, and Support Vector Machine, and models are evaluated using a classification report for best performance. The best performing classifier is used in the next part of the analysis to perform feature selection on the full dataset and each of the age group datasets. The top 15% of the most important features are selected from each dataset and compared to determine which attributes are most salient in classifying obesity. The full methodology can be simplified as:

1. Data Preprocessing:
    a. Data Exploration – Visualization of Key Features
    b. Feature Transformations
2. Cluster Analysis Exploration
3. Classification and Model Selection
4. Feature Selection
5. Results
6. Conclusion

Data Preprocessing

Data Exploration

Gender will be explored as visualization in with height and weight. In terms of height, male and female are similarly distributed according to the box plot in Figure A1.1 below. While males are generally taller than females, both male and female share a similar average in weight, with females having a much larger range of weight (as well as BMI) compared to male.

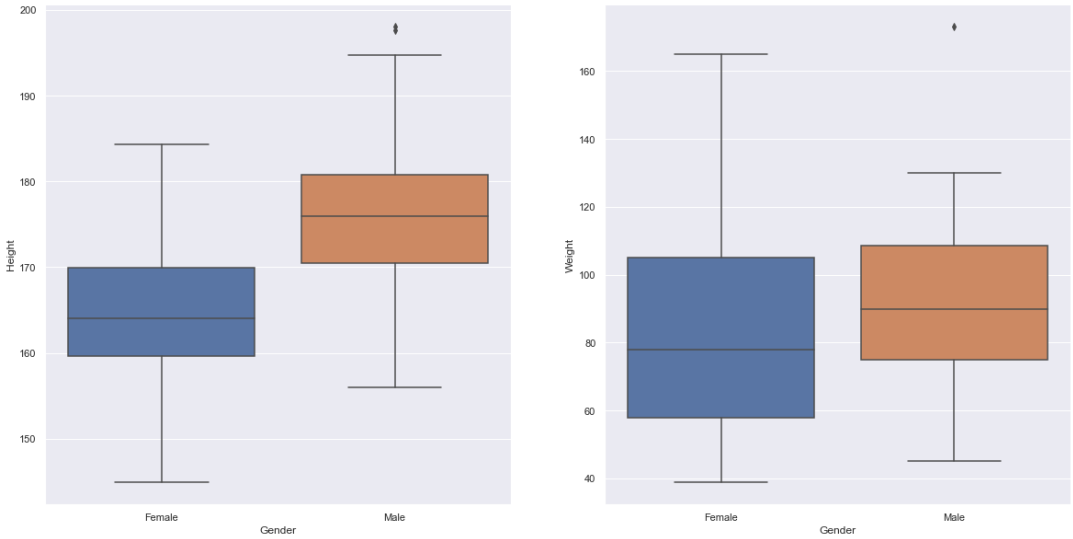


Figure A1.1 – Box Plot Comparing Male and Female Height and Weight (from Appendix A, Page 25)

Figure A1.2 below shows the Line plot between weight and height of females and males shows that the weight and height are more linear for females than males.

Figure A1.2 – Line Plot Comparing Gender to Weight and Height (from Appendix A, Page 27)

The class label obesity levels were also be visualized. Figure A1.3 shows the general pie chart distribution for the class label. It is almost equally distributed between all the elements of the obesity category types.

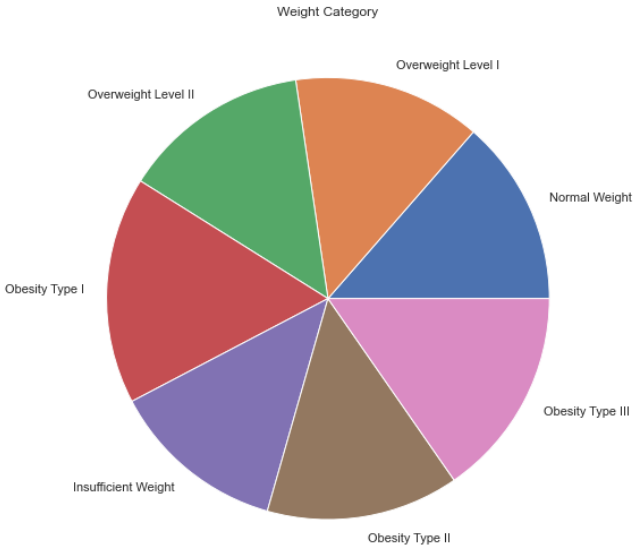


Figure A1.3 – Pie Chart of Obesity Type Distribution (from Appendix A, Page 28)

Figure A1.4 below shows that a bigger proportion of females with a higher BMI is reflected by the large slice of Obesity Type III in the pie chart below, while Obesity Type II is the most prevalent type of obesity in men. Interestingly, there is also a higher proportion of Insufficient Weight in females compared to male. These results could be explained by a heavier societal pressure on women in terms of dietary restrictions.

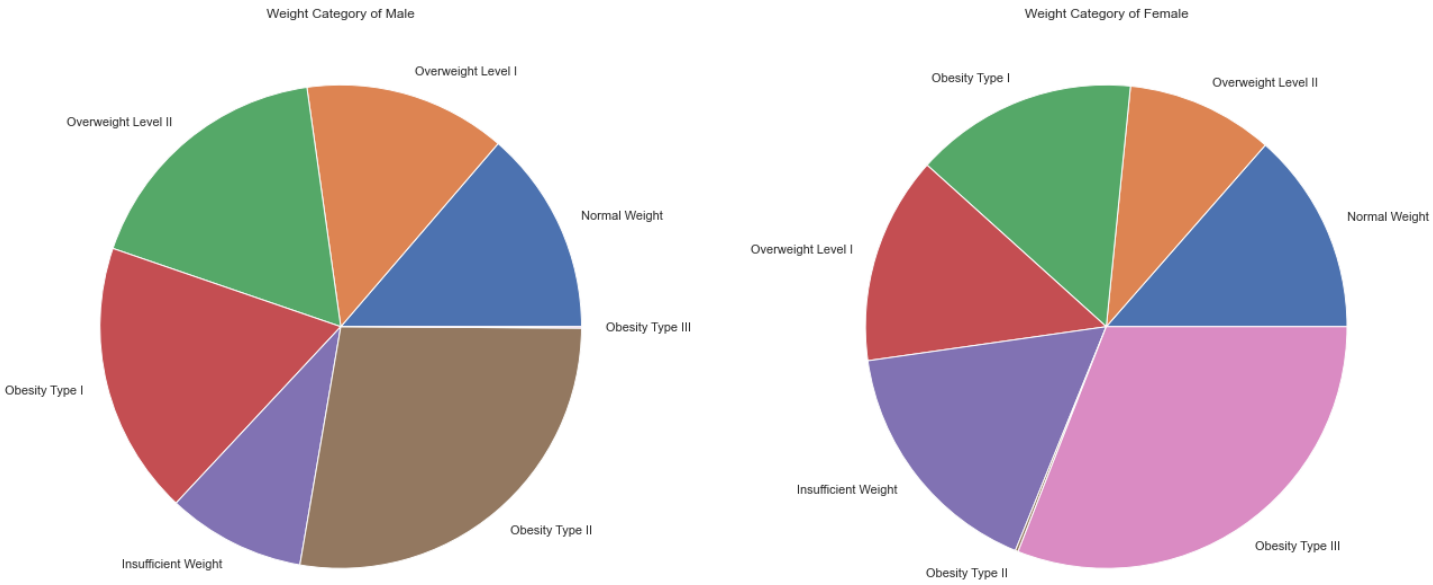


Figure A1.4 – Pie Charts Comparing Distribution of Obesity Types based on Gender (from Appendix A, Page 29)

Feature Transformations

In the original data set, there are numerical (both continuous and discrete) and categorical (including ordinal, non-ordinal, and binary features. Binary features are categorical features with a yes or no category. Imputation is used in both numerical and categorical data to fill in missing values. Feature scaling is employed for continuous numerical values, including age, weight, and height. Ordinal and Label Encoding are used for non-ordinal categorical data, such as means of transportation and obesity level, while One Hot Encoding is applied to data which is ordinal in nature (e.g., never, sometimes, always).

The above preprocessing procedures are bundled into a pipeline, which also applies multiple models on the data set in search for the best model. Since the classifier cannot operate with label data directly, One Hot Encoder and Label Encoding will be used to assign numeric values to each category. The class label, NObeyesdad, will be transformed into a digit label with LabelEncoder. StandardScaler is applied to attributes with values which ranges are not consistent with the rest, to avoid disproportionate weight assigned to these values (i.e., Age, Height, Weight).

Features that are ordinal in nature (i.e., answers including 'never', 'sometimes', 'always') will be preprocessed with OrdinalEncoder (this function is the same function as LabelEncoder, however LabelEncoder will take in multiple arguments as the latter is meant for target values only). Features that are non-ordinal in nature will be preprocessed with OneHotEncoder, so that the generated labels will not be interpreted in a way that suggests one answer is more important than the other (e.g., 3 is more important than 1). SimpleImputer is applied to all attributes to deal with missing values. All preprocessing techniques will be bundled into a pipeline, which will be deployed with the classifier models.

Cluster Analysis Exploration

Cluster analysis using K-Means algorithm was performed on the full dataset. To perform the cluster analysis, first, the class label, NObeyesdad is removed from the dataset. The dataset is transformed to ensure the correct data types exist for each feature. Dummy variables are created for the categorical features. The numeric dataset contains 2,111 rows and 43 columns. K-Means algorithm looks at the nearest neighbor based on distance to group datapoints into clusters. The standard Euclidean distance function is used for the K-means clustering. To ensure that the algorithm performs optimally, the data is scaled using min-max scaling. Min-max normalization scales all values of the data between 0 and 1. K-means requires selecting a number for K, which is the number of clusters. For this exploration, several values of K are explored including K = 5, K = 3, and K = 2. In addition to declaring the number for K, the algorithm also requires a stopping point, since the algorithm is designed to continue each iteration and repeat the clustering of the datapoints over and over. The maximum iterations used for this analysis is 500. For each value of K, the cluster centroids were examined to determine if any pattern exists in the data. A silhouette analysis is performed for to evaluate the separation between the resulting clusters and determine the quality of the clusters. The silhouette plots display a measure of how close each point in one cluster is to points in the neighboring clusters. The mean silhouette value is calculated and used as a threshold when determining the cluster quality. Clusters with most of their coefficients above the mean silhouette value are considered better quality which means that clusters are further away from the neighboring clusters. Clusters with most of their coefficients below the mean silhouette value reveals that samples are very close to the decision boundary between two neighboring clusters and negative coefficient values indicate that samples are assigned to the wrong cluster. When the silhouette plot does not

display any negative coefficients and have the thickest plots visually above the silhouette mean, the correct number of K has been selected.

Figure B1.1 below shows the results of the silhouette analysis for K = 5. The plot of the silhouettes shows that cluster 0 outperformed the other clusters with all its coefficients above the mean silhouette value. Cluster 4 also performed well with many of its coefficients above the mean silhouette value. The remaining three clusters did not perform as well since most of their coefficients are below the mean silhouette value. Four of the clusters display negative values with cluster 3 having the most negative coefficients, which indicates that 5 clusters are too high for the dataset.

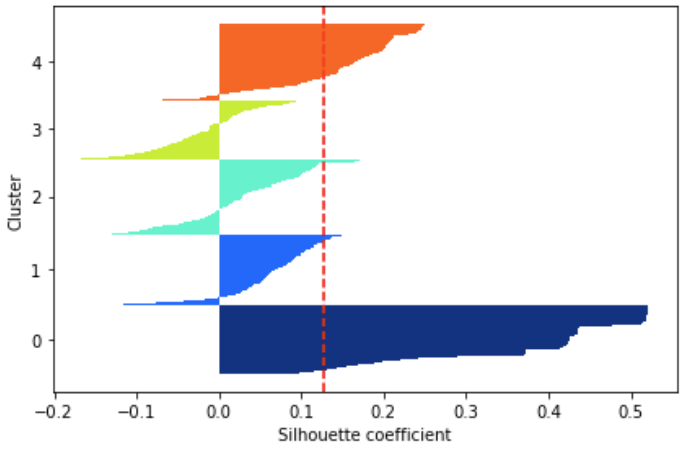


Figure B1.1 – K-Means at K=5, Silhouette Plot (from Appendix B, Page 56)

Next, K-means is performed again at K = 3. Figure B1.2 below shows the results of the silhouette analysis for K=3, which reveals that the algorithm performed neither better nor worse than at K = 5. The plot of the silhouettes shows that cluster 2 outperformed the other clusters with all its coefficients above the mean silhouette value. Cluster 1 performed the worst and did not have any coefficients above the mean silhouette value, but instead has negative coefficients. When evaluating the centroids, cluster 0 has Gender_Male with a value of 1.00 and Gender_Female with a value of 0. Cluster 0 most likely represents the male gender. Cluster 1 and 2 both contain a value of 0.99 for Gender_Female and 0.01 for Gender_Male, which shows that most likely Cluster 1 is misclassified. Most likely this cluster is pulling coefficients where it should not be and is too close to cluster 0 to be its own cluster. We can conclude from the silhouette plots that likely three cluster is still too high and that two clusters may be sufficient.

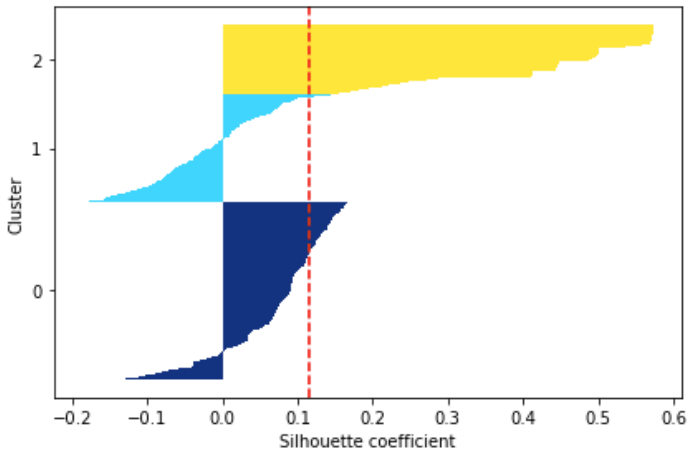


Figure B1.2 – K-Means at K=3, Silhouette Plot (from Appendix B, Page 60)

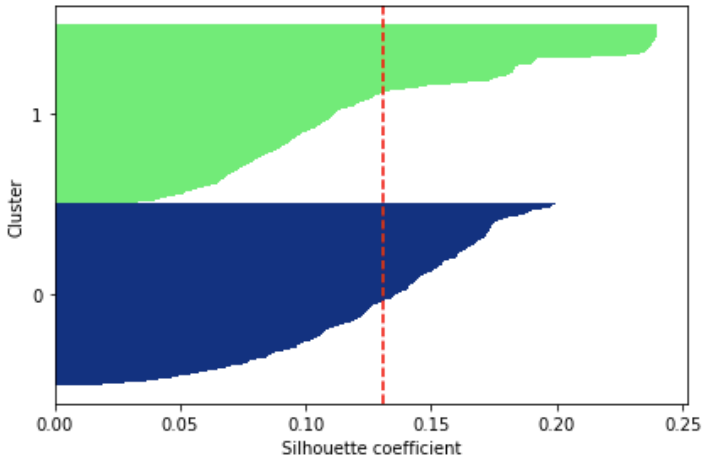


Figure B1.3 – K-Means at K=2, Silhouette Plot (from Appendix B, Page 65)

Lastly, K-means is performed again at K = 2. Figure B1.3 above shows the results of the silhouette analysis for K=2, which achieved the best silhouette plot compared to previous plots at K = 5 and K = 3. This silhouette plot shows that both cluster 0 and 1 have coefficients that are above the mean silhouette value and none of the coefficients are negative. Both clusters are neither thick nor full, although, cluster 0 appears thicker than cluster 1, but from the clustering results above, this result is most successful. When looking at the centroids, the two features that stand out that most likely represent the clusters compared to all other features is Gender_Male and Gender_Female. In cluster 0, Gender_Male has a value of 1.00 while Gender_Female has a value of -0.00 and in cluster 1, Gender_Female has a value of 1.00 while Gender_Male has a value 0.00. Moreover, we can conclude from the silhouette plots above that likely, cluster 0 represents males and cluster 1 represents female. This evaluation shows that a pattern exists by gender and that gender may play a role in the dataset and in determining classification of obesity levels.

Next, we will explore the K-means algorithm with the three generational age groups: Gen-Z, Millennials, and Gen-X and Boomers. This exploration is being explored to see if a pattern exists based on age range which the cluster analysis for the full dataset did not evaluate since the age groups were not grouped into categories. The youngest age is 14 and the oldest age is 61. The age groups are created by binning the Age attribute and then transforming the age group attribute into dummy variables. For exploratory purposes, K-means is performed on the dataset first without min-max normalization and second with min-max normalization at K = 3. The results of cluster analysis without normalization shows a very healthy silhouette plot with all three clusters full, thick, and with coefficients above the mean silhouette value. Figure B2.1 below confirms that clusters when age is grouped by range. When looking at the centroids, cluster 2 shows Gen-Z at 0.9 while Millennials at .10 and Gen-X and Boomers at 0.00. Most likely Gen-Z is represented in cluster 2.

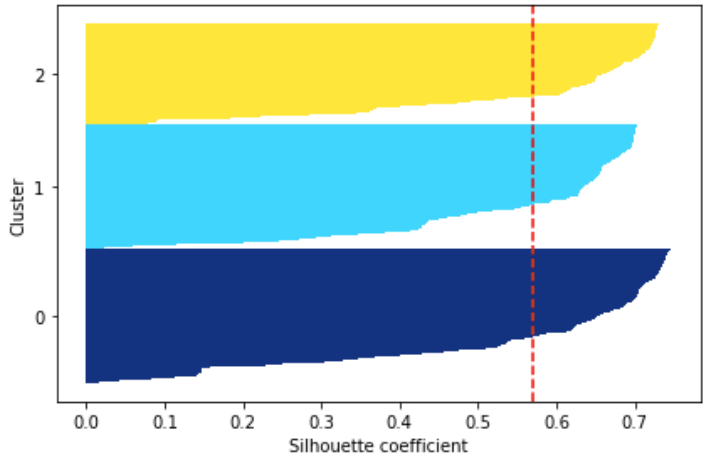


Figure B2.1 – K-Means at K=3 for Grouped Aged, Silhouette Plot (from Appendix B, Page 71)

The completeness and homogeneity scores were calculated for clusters since the class labels exist for further examination of the cluster quality. The completeness score was 0.70 which shows that members of a given class are assigned to the same cluster 70% of the time. The completeness score is positive and confirms that the clusters captured most of one class. The homogeneity score was much lower at 0.39 which shows that the clusters are not pure. These results may indicate that age group may be a factor in deciding the clusters for the data, but it may not be the main factor that affects obesity level for classification. The silhouette plots above display that a pattern exist but we must take into consideration that the data was not scaled. As such, we will next, perform K-means again with the data normalized to validate the results.

Figure B2.2 below shows the results silhouette analysis for K=2 with the normalized data. The results are drastically different from the results from Figure B2.1. Cluster 0 outperformed all other clusters with all its coefficients above the mean silhouette value. Cluster 2 performed adequately with many of its coefficients above the mean silhouette value and only a few of its coefficients in negative. Cluster 1 did not perform as well as many of the coefficients are in negative and none of them are above the mean silhouette value. When looking at the centroids, the values of the age group do not directly correspond to the silhouette plots.

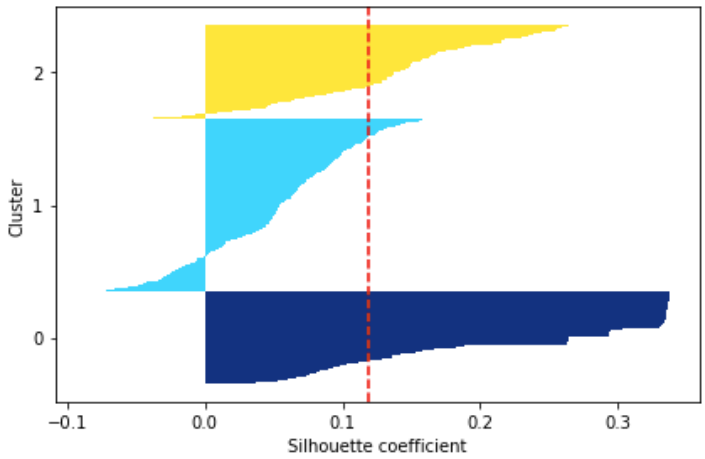


Figure B2.2 – K-Means at K=3 for Grouped Aged (normalized), Silhouette Plot (from Appendix B, Page 79)

These results show that with the normalized data, a pattern may not necessarily appear in the age groups. Moreover, when examining K-means and clustering, we can see how not scaling the data may lead to conclusions or patterns about the data when a pattern may not necessarily exist. This is validated when evaluating the completeness and homogeneity scores, which both resulted in low scores. The completeness score was around 0.34 and the homogeneity score is lower at 0.18. These scores show that grouping by age is not the main determining factor for the classification of obesity levels. Age still may play a role as a key feature, but the clustering exploration does not necessary reveal that the age groupings have a significant pattern. By building the classification models and performing feature selection, we will be able to obtain a better picture of age and age groupings and their role in classifying obesity levels.

Classification and Model Selection

Classification will be performed to predict discrete and nominal values (class and category labels) by organizing and categorizing data into different classes. For this analysis, the following classifiers are explored: K-Nearest Neighbor, Decision Tree, Stochastic Gradient Descent, and Support Vector Machines. The first step begins with the model construction where we construct a target function during training. The target will be the class and points will be class labels. The second step will be model evaluation, based on the test set. We will estimate the accuracy of the model. We will use a confusion matrix to evaluate the model accuracy. The final step will be classification to find or predict the outcomes for the actual class label (NObeyesdad) in an evaluation set.

To perform the classification, the classifiers are selected and stored in a list, each classifier will be looped through, and the preprocessor will be applied each time in the pipeline. The accuracy score of every classifier will be printed for comparison. The classification report is used to investigate the performance of each classifier in classes (type and level of obesity). 'Precision' shows the percentage of the classifier that can correctly predict the class (i.e., True Positive / (True Positive + False Positive). 'Recall' shows the percentage of the actual positive cases that the classifier can identify (i.e., True Positive / (True Positive + False Negative). 'F1' is the harmonic mean between Precision and Recall. 'Support' is the number of occurrences of the given class in the dataset. More consistent the amount of 'Support' of each class is, the more balanced the dataset. Detailed classification reports with all values for the classifiers are shown in the figures below:

Figure A2.1 - K-Nearest Neighbor Classifier Model (from Appendix A, Page 40):

Model Accuracy Score: 0.821

	precision		recall	f1-score	support
Insufficient Weight	0.72		0.92	0.81	25
Normal Weight	0.61		0.35	0.45	31
Obesity Type I	0.84		0.95	0.89	44
Obesity Type II	0.94		1.00	0.97	31
Obesity Type III	1.00		1.00	1.00	27
Overweight Level I	0.77		0.82	0.79	28
Overweight Level II	0.77		0.65	0.71	26
Accuracy				0.82	212
macro avg	0.81		0.81	0.80	212
weighted avg	0.81	0.82	0.81	212	

Figure A2.2 – Decision Tree Classifier Model (from Appendix A, Page 40):

Model Accuracy Score: 0.939

	precision	recall	f1-score	support
Insufficient Weight	0.96	0.92	0.94	25
Normal Weight	0.90	0.87	0.89	31
Obesity Type I	0.95	0.95	0.95	44
Obesity Type II	0.94	1.00	0.97	31
Obesity Type III	1.00	1.00	1.00	27
Overweight Level I	0.89	0.89	0.89	28
Overweight Level II	0.92	0.92	0.92	26
Accuracy			0.94	212
macro avg	0.94	0.94	0.94	212
weighted avg	0.94	0.94	0.94	212

Figure A2.3 – Stochastic Gradient Descent Classifier Model (from Appendix A, Page 42):

Model Accuracy Score: 0.575

	precision	recall	f1-score	support
Insufficient Weight	1.00	0.60	0.75	25
Normal Weight	0.30	0.94	0.45	31
Obesity Type I	0.92	0.25	0.39	44
Obesity Type II	0.82	1.00	0.90	31
Obesity Type III	1.00	1.00	1.00	27
Overweight Level I	0.37	0.25	0.30	28
Overweight Level II	0.50	0.08	0.13	26
Accuracy			0.58	212
macro avg	0.70	0.59	0.56	212
weighted avg	0.71	0.58	0.55	212

Figure A2.4 - C-Support Vector Classifier Model (C=0.025, probability=True) (from Appendix A, Page 40):

Model Accuracy Score: 0.505

	precision	recall	f1-score	support
Insufficient Weight	0.67	0.24	0.35	25
Normal Weight	0.35	0.19	0.25	31
Obesity Type I	0.34	1.00	0.51	44
Obesity Type II	0.83	0.77	0.80	31
Obesity Type III	1.00	1.00	1.00	27
Overweight Level I	0.00	0.00	0.00	28
Overweight Level II	0.00	0.00	0.00	26
Accuracy			0.50	212
macro avg	0.46	0.46	0.42	212
weighted avg	0.45	0.50	0.43	212

K-Nearest Neighbor (KNN) and Decision Tree Classifier models are the top two models that score the highest in terms of accuracy. KNN score 0.821 and Decision Tree scored 0.939 model accuracy values.

Feature Selection

The best classifier model for the dataset was Decision Tree. The Decision Tree classifier model will be used to evaluate the full dataset and each age-group dataset and perform feature selection. Decision Trees are non-parametric supervised learning algorithms used for both classification and regression. The models predict the value of a target variable by learning rules that the model creates from the dataset. Decision Tree models are considered best for data with high categorical variables and has the advantage of being able to perform feature selection. Feature selection is a dimensionality reduction method that select key features and improves a models accuracy score or retain the model's accuracy score while limiting the number of features. Feature selection uses a statistic test to test the samples and retrieve a specified number of features. In this analysis, the chi-square test is used as the statistic measure which is a common statistical approach for categorical features. Feature selection will be used to determine the top 15% of features for the data. This percentage was selected as it returned seven features which is a better number of features for interpretation.

The feature selection process begins by splitting the data to training set, training labels, testing set, and testing labels using an 80/20 random split with an 80% training set and a 20% testing set. The Decision Tree classifier model is then trained on the training set. Then, the model predicts on the test set. The classification report is generated showing the model accuracy and the confusion matrix. Then feature selection is performed with chi-square at a percentile of 15% and the resulting features are then transformed with the training set. Lastly, the model is retrained using the transformed training set with top features. The testing set is also transformed using the top features. Thereafter, the transformed training and testing sets are then evaluated with decision tree to reveal the accuracy of the transformed model. The accuracy is compared, and conclusions can be made regarding the top features.

The classifier model on the full dataset performed very well with an accuracy of 94.1%. Class 4: 'Obesity_Type_III' had a 100% accurate prediction. Class 0: 'Insufficient_Weight' and 3: 'Obesity_Type_II' achieved above 95% accuracy. Class 6: 'Obesity_Type_II' had the lowest accuracy at 91%. The accuracy for the training set is 100% and the accuracy for the test set is 94.09%. The model is performing well and not overfitting since the accuracy for the test set is very close to the training set and not experiencing high variance. With the feature selection, using the top 15% of features, the classifier still performed well with an accuracy of 86.3%. Although, the accuracy reduced from the original feature set, the reduced feature set contains only seven features and still achieved a high level of accuracy. Class 1: 'Normal Weight' and Class 6: 'Overweight_level II' had the lowest accuracy score at 74% and 75% respectively. Class 4: 'Obesity Type III' achieved 100% accuracy and Class 3: 'Obesity Type II' still maintained over 95% accuracy. Moreover, for the full dataset, the top features that are associated to obesity levels are Age, Weight, Gender_Female, Gender_Male, family_history_with_overweight as 'no', FCVC as 'Always' and CAEC as 'Frequently.' These results confirm that male and female genders are salient features for the classification of obesity levels. This result mirrors the results of the cluster analysis exploration which split the data into two clusters representing male and female genders. Besides gender, the results also show that individuals indicating no history of obesity in their family as an important feature when classifying obesity levels. This shows that hereditary, family, or environmental factors associated with families with a history of obesity, plays a key role in an individual's obesity levels. Lastly, two eating habit features, always eating vegetables with meals (FCVC) and frequently eating food between meals round up the top features. Surprisingly, daily activity features and physical activity features were not included in the top features. instead, biological factors and eating habits were features that had more precedents in determining obesity levels.

The classifier model on the Gen-Z dataset performed equally as well as the full dataset with slightly lower accuracy at 91.9%. Class 0: 'Insufficient_Weight' and 2: 'Obesity_Type_I' achieved above 95% accuracy. Class 3: 'Obesity_Type_II' had the lowest accuracy at 84%. Class 1: 'Normal_Weight' and 6: 'Overweight_Level_II' had the next lowest accuracy at 88% and 89% respectively. Moreover, with the Gen-Z dataset, the model performed better in prediction of Class 0: 'Insufficient_Weight' and 2: 'Obesity_Type_I'. Both the full dataset and the Gen-Z dataset had lowest accuracy with Class 6: 'Overweight_Level_II'. With feature selection using the top 15% of features, the accuracy of the classifier decreased to 80.8%. This model with feature selection does not perform as well as the model using the full dataset. Class 4 had the highest accuracy at 98%, which is comparable to the full dataset which predicted class 4 at 100%. Class 6: 'Overweight_Level II' had the lowest accuracy at 58%. This shows that for the Gen-Z age group, the model is unable to classify 'Overweight_Level II' using the top 15% of features. Likely, this means that other attributes are required to accurately classify this obesity level. Like the model using the full dataset, this model also does not classify Class 1: 'Normal_Weight' or Class 3: 'Obesity_Type_II' as well as the other classes. In both models, the lowest accuracy was in classifying class 1: 'Normal_Weight'. The lack of accuracy in predicting normal weight could be due to how the features were crafted for the data. The features were created to help determine if an individual is overweight and not necessarily to help determine normal weight. Likely, this means that these features are not salient to classifying an individual that is neither underweight nor overweight. The top 15% of features includes weight, gender_male, family_history_with_obesity as ‘no’, always eating vegetables with meals (FCVC) as ‘always’, and frequently eating food between meals. These features are the same top features from the model using the full dataset except, for gender only male gender is included. Two additional eating habits features are included in the top features with the Gen-Z age group: not eating high calorie foods frequently and number of meals consumed daily. Moreover, eating habit features are the most important features in association with obesity level for Gen-Z age group along with biological and hereditary features. Like the model in the full dataset, physical activity features were not included in the top features for the classification of obesity levels.

The classifier model on the Millennials dataset did not perform as well as the model for the Gen-Z or full dataset. The model achieved an accuracy of 89.6%. Like the two previous models, Class 4: 'Obesity_Type_III' had a prediction accuracy of 100%. Unlike the two previous models, Class 6: 'Overweight_Level II' and Class 3: 'Obesity_Type_II' performed better in this model with an accuracy of 95%. Class 0: 'Insufficient Weight' had an accuracy of 67%, which is starkly lower in accuracy compared to the previous two models. Class 1: 'Normal_Weight' also had a low accuracy at 71%. This aligns with the two previous models, which also had the lowest accuracy in predicting Class 1: 'Normal_Weight'. With the feature selection using the top 15% of features, the model’s accuracy dropped to 79.9%. The model performs slightly worse than the model for Gen-Z age group. Class 4 again, had the highest accuracy at 100%, which is comparable to the full dataset which also predicted class 4 at 100%. Class 2: 'Obesity_Type_I' and Class 5: 'Overweight_Level_I' had the lowest accuracy at 65%. Class 6: 'Overweight_Level_II' has a significant drop in accuracy, which prior to feature selection had a 95% prediction, and after feature selection has a 68% prediction. This shows that the features necessarily to predict Class 6 are not included in the top 15% features. The model also does not classify Class 1: 'Normal_Weight' as well as the other classes, which is consistent pattern among all the models. In contrast, the model was able to predict Class 3: 'Obesity_Type_II' better than the model for the Gen-Z age group. The top 15% of features includes weight, gender both male and female, family_history_with_ obesity as ‘no’, always eating vegetables with meals (FCVC), and frequently eating food between meals. These features are the same top features from the model using the full dataset. Unlike the previous two models, this model includes one additional top feature, a physical activity feature, means of transportation as automobile. This is interesting since previous models did not include a physical activity feature. Moreover, the model with

the top 15% features for both the Millennials age group and the Gen-Z age group yielded similar accuracy for classification. The main difference is that a physical activity feature is included in the top features for Millennials which is not included for Gen-Z.

The classifier model for the Gen-X and Boomers dataset performed the worse compared to all previous models. The model achieved an accuracy of 66.7%. This model resulted in the lowest accuracy score compared to the previous models. This dataset is significantly smaller than the previous two dataset. As such, not all classes are represented in this model and due to the limited number of entries, the model does not have as much data for the classifier to train on compared to previous three models. This model was able to predict Class 1: 'Normal_Weight' at 89% accuracy, which is higher in accuracy compared to all previous models. This model was unable to predict Class 0: 'Insufficient_Weight or Class 3: 'Obesity_Type_II'. With the feature selection using the top 15% of features, the model maintained its accuracy at 66.7%. This model underperformed compared to all previous models with all classes having accuracy scores of 75% or lower. Again, the model was unable to predict Class 0: ‘Insufficient_Weight.’ Since some classes are not represented in this dataset and with a lower amount of data for training, it is not unexpected that the model was unable to classify obesity levels as well as the previous models. The top 15% of features includes weight and always eating vegetables with meals (FCVC) which are two features also included as top features for the full dataset, Gen-Z dataset, and Millennial’s dataset. Additional eating habits features are included as top features: water intake at more than 2 liters per day and monitoring calories intake daily. In addition, physical activity features include direct physical activity 1 to 2 days or 3 to 4 days and means of transportation by public transit. This is interesting since previous models did not include specific eating habit features such as water intake and direct exercise or direct physical activity. The results are drastically different from the Gen-Z and Gen-X dataset but since the sample size is significantly lower, more data would be needed for this population to perform a more detailed and thorough analysis in validating these top features and determining what key features affect the classification of obesity for the Gen-X and Boomers age group.

Results

Two classifiers performed well for classification of obesity levels: Decision Tree and K-Nearest Neighbor (KNN). Since the dataset contained many categorical variables, the classifier model using Decision Tree resulted in the highest accuracy. Feature selection was performed using Decision Tree to determine the top 15% most important features. With feature selection, the model using the full dataset resulted in the best accuracy. The top 15% features for this model include age, weight, male gender, no family history with obesity, always eating vegetables with meals (FCVC) and frequently eating food between meals (CAEC). With feature selection, the model still performed well with an accuracy of 86.3%. Biological features and family history with obesity are top features that are associated with classifying obesity. With the full dataset, only two additional eating habit features were top features. The models for Gen-Z age group and Millennials age group also included weight, both male and female gender, no family history with obesity, always eating vegetables with meals (FCVC), and frequently eating food between meals (CAEC) as top features. Gen-Z includes more eating habit features including not eating high calorie foods frequently and number of meals consumed daily, and Millennials includes a physical activity feature which is transportation by automobile. The model for the Gen-X and Boomers age group performed the worst and had different top features compared to all other models. The top features still included weight, but no longer included gender and instead included both eating habits and physical activity features including water

intake of 2 liters or more, calories intake daily, and direct physical activity. The model had significantly lower amount of data compared to previous models which may contribute to the lower accuracy and lower performance.

The cluster analysis showed a pattern with gender but did not necessarily show a pattern with age groups. Although from the feature selection, age plays a salient factor in classifying obesity level, the generational age groupings used for this analysis may not necessarily be as significant. The results conclude that eating habit features such as always eating vegetables with meals and frequently eating between meals are salient factors in determining obesity levels among younger adults. Since the data for the older adults were conclusive due to low sample size, we cannot confidently specify which eating habit or physical activity features are salient factors in determining obesity levels for older adults. We can conclude that biological factors such as gender, height, weight, and no family history of obesity play an important role in classifying obesity levels. Contrary to popular belief, physical activity features were not the most important features in classifying obesity level. Since the class label using the body mass index, which is an index based on nutrition, the outcome confirms that ultimately diet and nutrition is the key to classifying obesity levels.

Conclusion

In this analysis, clustering analysis was explored on the dataset to discover patterns by groups in the data. Although age was grouped, a pattern was only found with gender and not with age. Classification and model selection was performed, and Decision Tree was selected as the best classifier. Feature selection was performed using the classifier model to determine the top 15% of features which represents the most salient factors in determining obesity levels among adults in regions of Latin America. This analysis of the dataset containing obesity levels among adults from Mexico, Peru, and Colombia revealed that factors that affect the classification of obesity levels the most are age, gender, weight, and family history with obesity. The original prediction stated that eating high calorie foods frequently, eating between meals, and having lower days of physical activity would be the most salient attributes in determining obesity levels. However, the results show that always eating vegetables with meals and frequently eating between meals are the most important factors besides gender, weight, and family history with obesity. Physical activity attributes were not the most important factors in classifying obesity levels. Although age is an important factor, an individual's generational group does not play a salient factor in classifying obesity level. Moreover, obesity level as defined here is a measure of nutritional status and thus, diet and eating habit factors are more salient for classification. Due to the low sample size of older adults, future studies could focus on studying the older adult population in greater depth. In addition, this analysis did not focus on analyzing male and females separately. Future analysis can separate the two groups and perform separate analysis on the two types of gender since often gender differences may reveal latent factors in the data.

Works Cited

World Health Organization. “Obesity and Overweight.” World Health Organization, World Health Organization: WHO, 9 June 2021, www.who.int/news-room/fact-sheets/detail/obesity-and-overweight.

“Body Mass Index - BMI.” World Health Organization, World Health Organization, 20 Nov. 2021, https://www.euro.who.int/en/health-topics/disease-prevention/nutrition/a-healthy-lifestyle/body-mass-index-bmi.

Kain J, Viol F, Albala C. Obesity trends and determinant factors in Latin America. Cad Saude Publica. 2003;19 Suppl 1: S77-86. doe: 10.1590/s0102-311x2003000700009. Epub 2003 Jul 21. PMID: 12886438.

Palechor, F. M., & de la Hoz Manotas, A. (2019). Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru, and Mexico. Data in Brief, 104344.

Appendix A: Preprocessing and Classification

Importall the required libraries

In [6]:
```
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import numpy as np
import collections
from collections import Counter

import sklearn
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import SGDClassifier

from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

Reading File in to a DataFrame

In [7]:
```
df = pd.read_csv('/Users/kokilamaddi/Documents/Final Assignment/ObesityDataSet/O
```

In [8]:
```
df
```

Out[8]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	N
0	Female	21.000000	1.620000	64.000000	yes	no	2.0	
1	Female	21.000000	1.520000	56.000000	yes	no	3.0	
2	Male	23.000000	1.800000	77.000000	yes	no	2.0	
3	Male	27.000000	1.800000	87.000000	no	no	3.0	
4	Male	22.000000	1.780000	89.800000	no	no	2.0	
...	...	...	...	...	...	...	...	
2106	Female	20.976842	1.710730	131.408528	yes	yes	3.0	
2107	Female	21.982942	1.748584	133.742943	yes	yes	3.0	
2108	Female	22.524036	1.752206	133.689352	yes	yes	3.0	

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	N(
2109	Female	24.361936	1.739450	133.346641	yes	yes	3.0	3
2110	Female	23.664709	1.738836	133.472641	yes	yes	3.0	3

2111 rows × 17 columns

In [9]:
```
df.shape
```

Out[9]: (2111, 17)

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Gender                          2111 non-null   object 
 1   Age                             2111 non-null   float64
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   family_history_with_overweight  2111 non-null   object 
 5   FAVC                            2111 non-null   object 
 6   FCVC                            2111 non-null   float64
 7   NCP                             2111 non-null   float64
 8   CAEC                            2111 non-null   object 
 9   SMOKE                           2111 non-null   object 
 10  CH2O                            2111 non-null   float64
 11  SCC                             2111 non-null   object 
 12  FAF                             2111 non-null   float64
 13  TUE                             2111 non-null   float64
 14  CALC                            2111 non-null   object 
 15  MTRANS                          2111 non-null   object 
 16  NObeyesdad                      2111 non-null   object 
dtypes: float64(8), object(9)
memory usage: 280.5+ KB
```

In [11]:
```
df.describe()
```

Out[11]:

	Age	Height	Weight	FCVC	NCP	CH2O	FAF
count	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000
mean	24.312600	1.701677	86.586058	2.419043	2.685628	2.008011	1.010298
std	6.345968	0.093305	26.191172	0.533927	0.778039	0.612953	0.850592
min	14.000000	1.450000	39.000000	1.000000	1.000000	1.000000	0.000000
25%	19.947192	1.630000	65.473343	2.000000	2.658738	1.584812	0.124505
50%	22.777890	1.700499	83.000000	2.385502	3.000000	2.000000	1.000000
75%	26.000000	1.768464	107.430682	3.000000	3.000000	2.477420	1.666678
max	61.000000	1.980000	173.000000	3.000000	4.000000	3.000000	3.000000

In [12]:
```
df.columns
```

Out[12]:
```
Index(['Gender', 'Age', 'Height', 'Weight', 'family_history_with_overweight',
       'FAVC', 'FCVC', 'NCP', 'CAEC', 'SMOKE', 'CH2O', 'SCC', 'FAF', 'TUE',
       'CALC', 'MTRANS', 'NObeyesdad'],
      dtype='object')
```

In [13]:
```
df.columns = ['Gender', 'Age', 'Height', 'Weight', 'family_history_with_overweig
              'SMOKE', 'CH2O', 'SCC', 'FAF', 'TUE','CALC', 'MTRANS', 'NObeyesdad

df
```

Out[13]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	N
0	Female	21.000000	1.620000	64.000000	yes	no	2.0	
1	Female	21.000000	1.520000	56.000000	yes	no	3.0	
2	Male	23.000000	1.800000	77.000000	yes	no	2.0	
3	Male	27.000000	1.800000	87.000000	no	no	3.0	
4	Male	22.000000	1.780000	89.800000	no	no	2.0	
...	...	...	...	...	...	...	...	
2106	Female	20.976842	1.710730	131.408528	yes	yes	3.0	
2107	Female	21.982942	1.748584	133.742943	yes	yes	3.0	
2108	Female	22.524036	1.752206	133.689352	yes	yes	3.0	
2109	Female	24.361936	1.739450	133.346641	yes	yes	3.0	
2110	Female	23.664709	1.738836	133.472641	yes	yes	3.0	

2111 rows × 17 columns

In [14]:
```
df['NObeyesdad'] = df['NObeyesdad'].apply(lambda x: x.replace('_', ' '))
df['MTRANS'] = df['MTRANS'].apply(lambda x: x.replace('_', ' '))
df['Height'] = df['Height']*100
df['Height'] = df['Height'].round(1)
df['Weight'] = df['Weight'].round(1)
df['Age'] = df['Age'].round(1)
df
```

Out[14]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAI
0	Female	21.0	162.0	64.0	yes	no	2.0	3.0	Sometim
1	Female	21.0	152.0	56.0	yes	no	3.0	3.0	Sometim
2	Male	23.0	180.0	77.0	yes	no	2.0	3.0	Sometim
3	Male	27.0	180.0	87.0	no	no	3.0	3.0	Sometim
4	Male	22.0	178.0	89.8	no	no	2.0	1.0	Sometim

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAI
...	...	...	...	...	...	...	...	...	
2106	Female	21.0	171.1	131.4	yes	yes	3.0	3.0	Sometim
2107	Female	22.0	174.9	133.7	yes	yes	3.0	3.0	Sometim
2108	Female	22.5	175.2	133.7	yes	yes	3.0	3.0	Sometim
2109	Female	24.4	173.9	133.3	yes	yes	3.0	3.0	Sometim
2110	Female	23.7	173.9	133.5	yes	yes	3.0	3.0	Sometim

2111 rows × 17 columns

In [15]:
```
for x in ['FCVC', 'NCP', 'CH2O', 'FAF', 'TUE']:
    value = np.array(df[x])
    print(x,':', 'min:', np.min(value), 'max:', np.max(value))
```

```
FCVC : min: 1.0 max: 3.0
NCP : min: 1.0 max: 4.0
CH2O : min: 1.0 max: 3.0
FAF : min: 0.0 max: 3.0
TUE : min: 0.0 max: 2.0
```

Appendix A1

Exploratory Data Analysis

In [16]:
```
for x in ['FCVC', 'NCP', 'CH2O', 'FAF', 'TUE']:
    df[x] = df[x].apply(round)
    value = np.array(df[x])
    print(x,':', 'min:', np.min(value), 'max:', np.max(value), df[x].dtype)
    print(df[x].unique())
```

```
FCVC : min: 1 max: 3 int64
[2 3 1]
NCP : min: 1 max: 4 int64
[3 1 4 2]
CH2O : min: 1 max: 3 int64
[2 3 1]
FAF : min: 0 max: 3 int64
[0 3 2 1]
TUE : min: 0 max: 2 int64
[1 0 2]
```

In [17]:
```
df1 = df.copy()
```

In [18]:
```
mapping0 = {1:'Never', 2:'Sometimes', 3:'Always'}
mapping1 = {1: '1', 2:'2' , 3: '3', 4: '3+'}
mapping2 = {1: 'Less than a liter', 2:'Between 1 and 2 L', 3:'More than 2 L'}
```

```
mapping3 = {0: 'I do not have', 1: '1 or 2 days', 2: '2 or 4 days', 3: '4 or 5 d
mapping4 = {0: '0–2 hours', 1: '3–5 hours', 2: 'More than 5 hours'}
```

In [19]:

```
df['FCVC'] = df['FCVC'].replace(mapping0)
df['NCP'] = df['NCP'].replace(mapping1)
df['CH2O'] = df['CH2O'].replace(mapping2)
df['FAF'] = df['FAF'].replace(mapping3)
df['TUE'] = df['TUE'].replace(mapping4)
```

In [20]:

```
df
```

Out[20]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	
0	Female	21.0	162.0	64.0	yes	no	Sometimes	3	Son
1	Female	21.0	152.0	56.0	yes	no	Always	3	Son
2	Male	23.0	180.0	77.0	yes	no	Sometimes	3	Son
3	Male	27.0	180.0	87.0	no	no	Always	3	Son
4	Male	22.0	178.0	89.8	no	no	Sometimes	1	Son
...	...	...	...	...	...	...	...	...	
2106	Female	21.0	171.1	131.4	yes	yes	Always	3	Son
2107	Female	22.0	174.9	133.7	yes	yes	Always	3	Son
2108	Female	22.5	175.2	133.7	yes	yes	Always	3	Son
2109	Female	24.4	173.9	133.3	yes	yes	Always	3	Son
2110	Female	23.7	173.9	133.5	yes	yes	Always	3	Son

2111 rows × 17 columns

Age, Height and Weight

In terms of height, male and female are similarly distributed according to the box plot below. While male are generally taller than females, both male and female share a similar average in weight, with females having a much larger range of weight (as well as BMI) compared to male. This is further illustrated by the steeper line plot between weight and height of females than male.

Figure A1.1

In [21]:
```
sns.set()
fig = plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
sns.boxplot(x='Gender', y='Height', data=df)
plt.subplot(1, 2, 2)
sns.boxplot(x='Gender', y='Weight', data=df)
```

Out[21]: <AxesSubplot:xlabel='Gender', ylabel='Weight'>

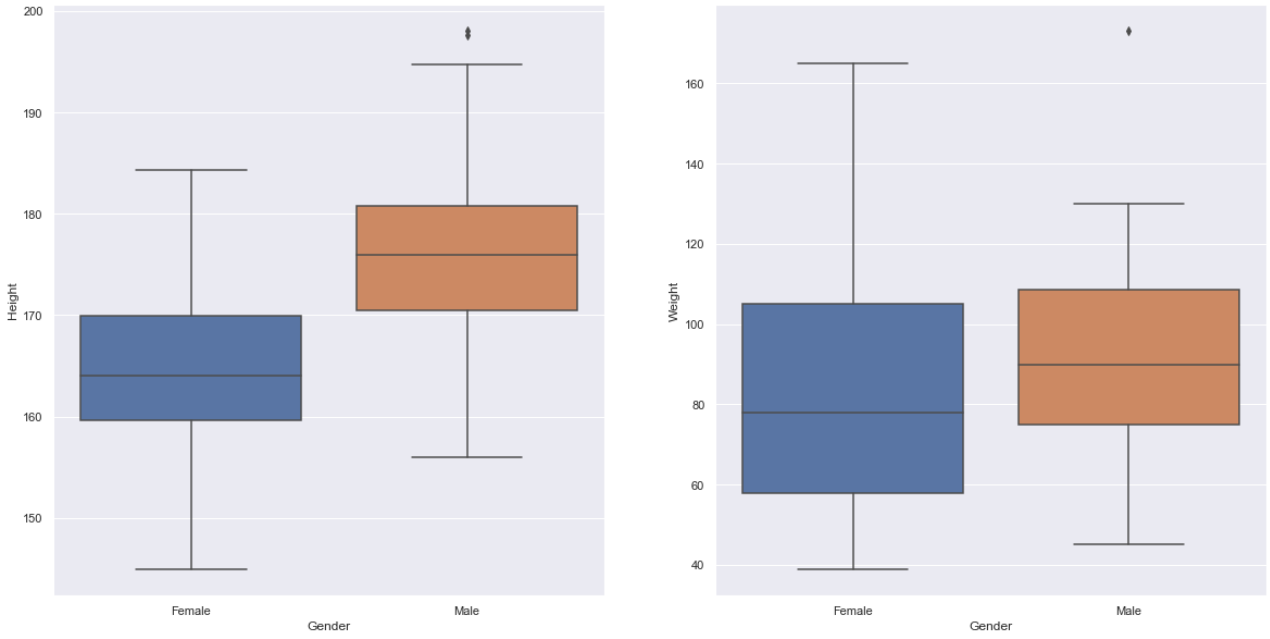


Figure A1.1

In [22]:
```
sns.set()
g = sns.jointplot("Height", "Weight", data=df,
                  kind="reg", truncate=False,
                  xlim=(125, 200), ylim=(35, 180),
                  color="m", height=10)
g.set_axis_labels("Height (cm)", "Weight (kg)")
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[22]: <seaborn.axisgrid.JointGrid at 0x7fa739b244f0>

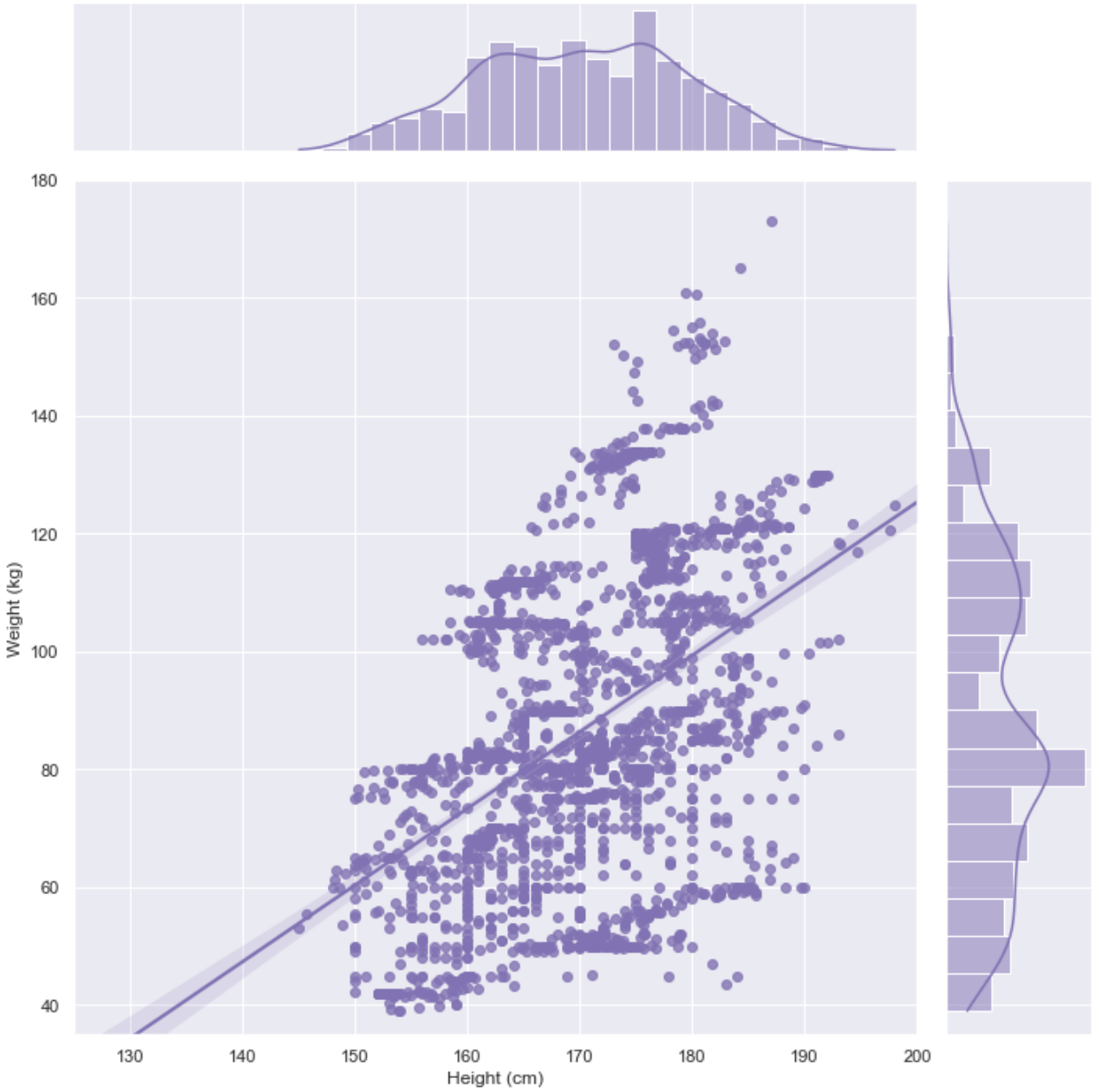


Figure A1.2

In [23]:
```
g = sns.lmplot(x="Height", y="Weight", hue="Gender",
               height=10, data=df)
g.set_axis_labels("Height (cm)", "Weight (kg)")
```

Out[23]: <seaborn.axisgrid.FacetGrid at 0x7fa73d620640>

Figure A1.2

Obesity

In [24]:
```
c = Counter(df['NObeyesdad'])
print(c)
```

```
Counter({'Obesity Type I': 351, 'Obesity Type III': 324, 'Obesity Type II': 297,
'Overweight Level I': 290, 'Overweight Level II': 290, 'Normal Weight': 287, 'In
sufficient Weight': 272})
```

Figure A1.3

In [25]:
```
fig = plt.figure(figsize=(8,8))
plt.pie([float(c[v]) for v in c], labels=[str(k) for k in c], autopct=None)
plt.title('Weight Category')
plt.tight_layout()
```

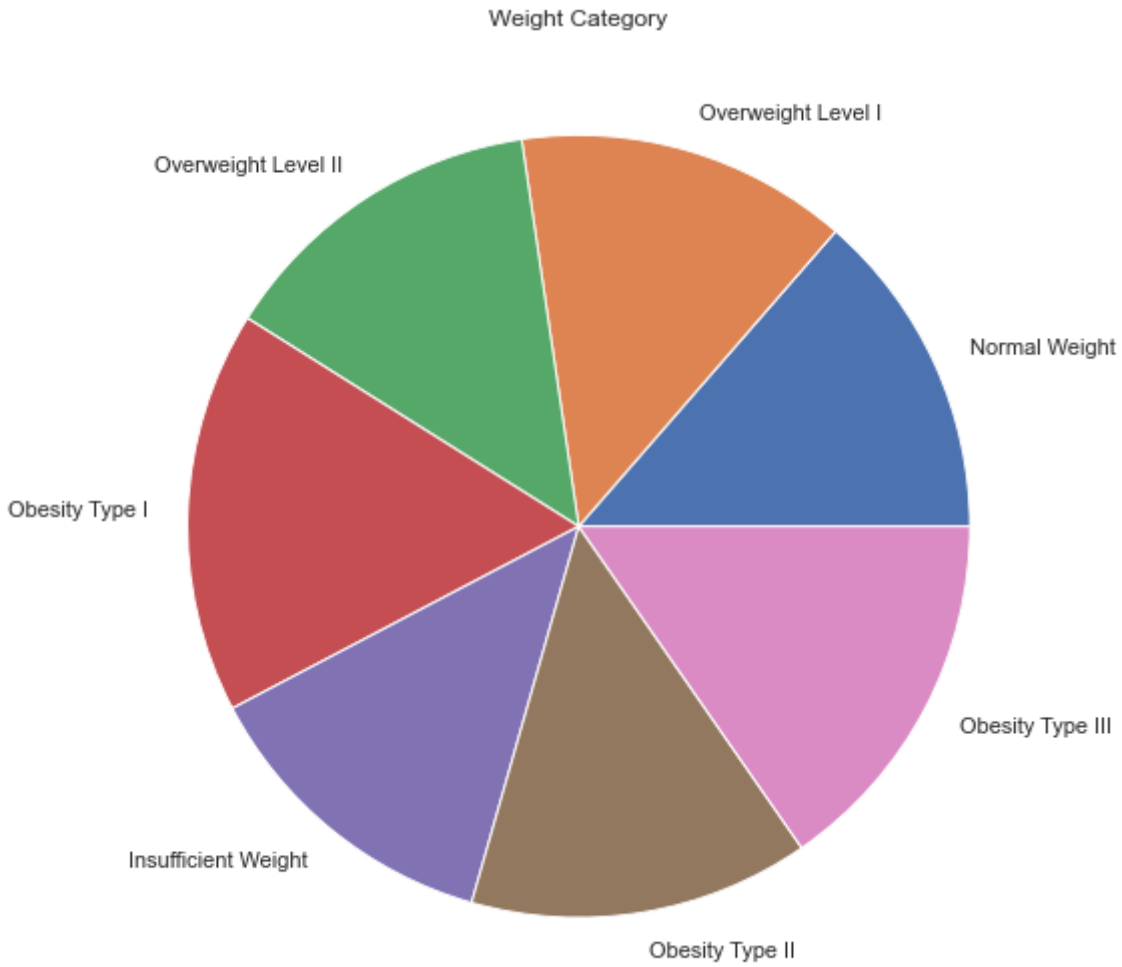


Figure A1.3

In [27]:
```
filt = df['Gender'] == 'Male'
c_m = Counter(df.loc[filt, 'NObeyesdad'])
print(c_m)
c_f = Counter(df.loc[~filt, 'NObeyesdad'])
print(c_f)
```

```
Counter({'Obesity Type II': 295, 'Obesity Type I': 195, 'Overweight Level II': 1
87, 'Normal Weight': 146, 'Overweight Level I': 145, 'Insufficient Weight': 99,
'Obesity Type III': 1})
Counter({'Obesity Type III': 323, 'Insufficient Weight': 173, 'Obesity Type I':
156, 'Overweight Level I': 145, 'Normal Weight': 141, 'Overweight Level II': 10
3, 'Obesity Type II': 2})
```

A bigger proportion of female with a higher BMI is reflected by the large slice of Obesity Type III in the pie chart below, while Obesity Type II is the most prevalent type of obesity in make. Interestingly, there is also a higher proportion of Insufficient Weight in female compared to male, this could be explained by a heavier societal pressure on women to go on diets.

Figure A1.4

In [28]:
```
fig = plt.figure(figsize=(20,8))
plt.subplot(1, 2, 1)
plt.pie([float(c_m[v]) for v in c_m], labels=[str(k) for k in c_m], autopct=None
plt.title('Weight Category of Male')
```

```
plt.tight_layout()

plt.subplot(1, 2, 2)
plt.pie([float(c_f[v]) for v in c_f], labels=[str(k) for k in c_f], autopct=None
plt.title('Weight Category of Female')
plt.tight_layout()
```

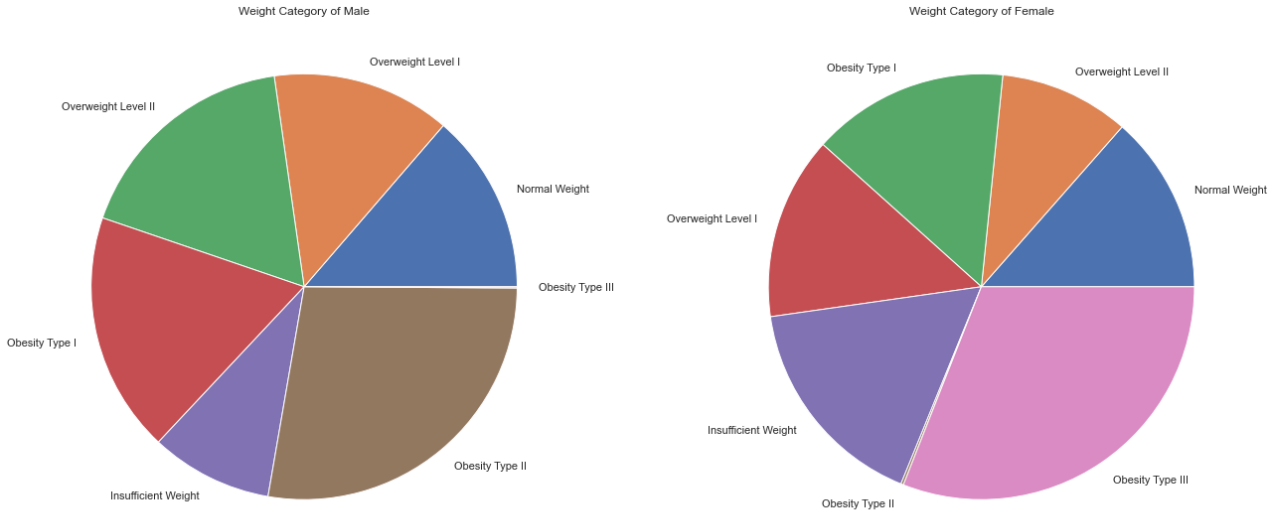


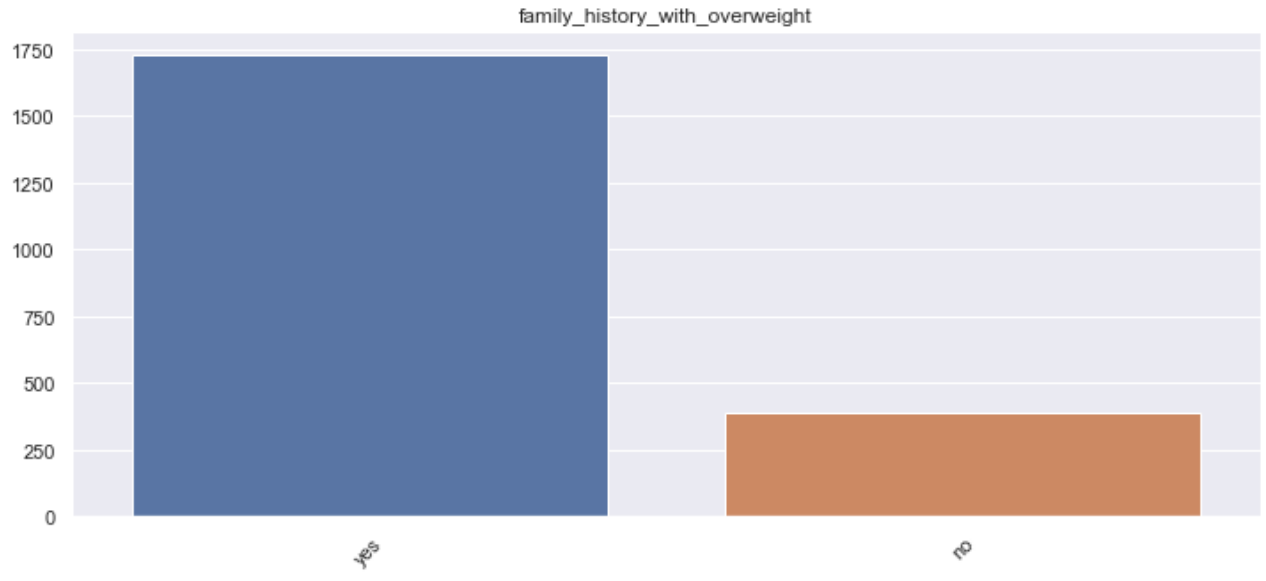
Figure A1.4

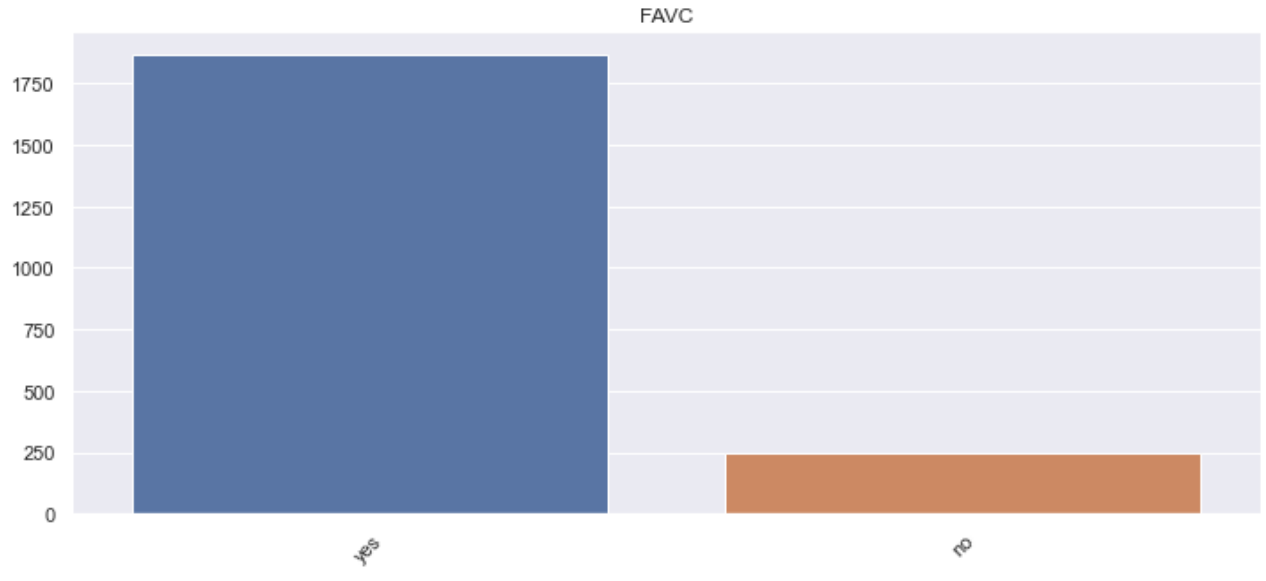
Eating and Exercise Routine

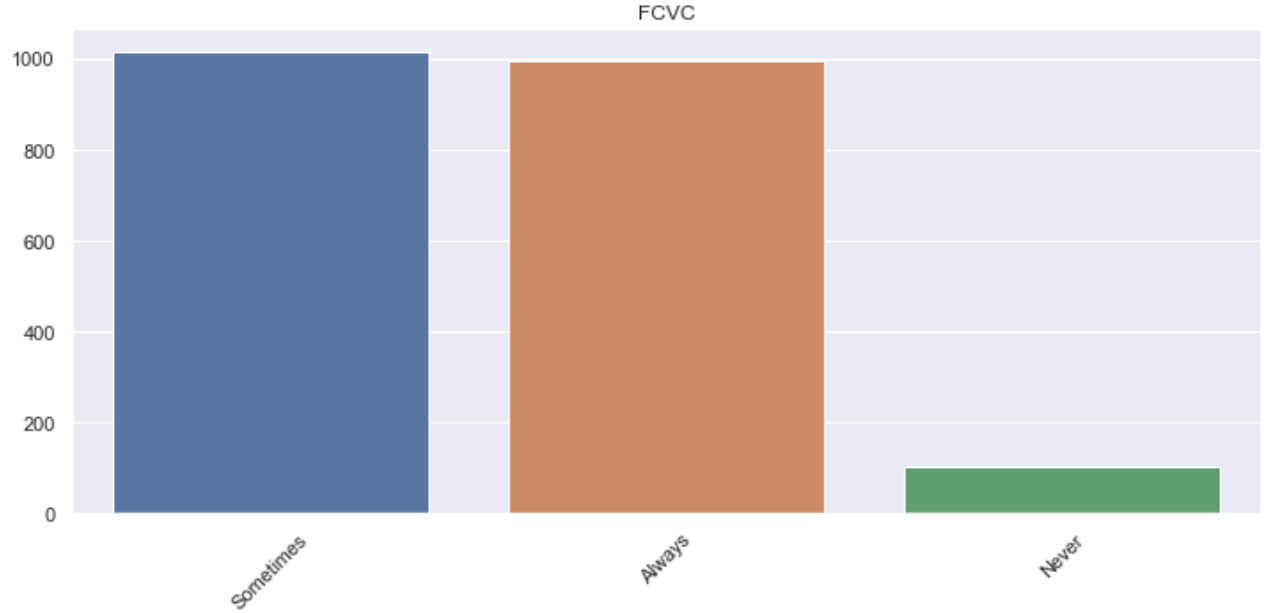
In [29]:

```
for a in df.columns[4:-1]:
    data = df[a].value_counts()
    values = df[a].value_counts().index.to_list()
    counts = df[a].value_counts().to_list()

    plt.figure(figsize=(12,5))
    ax = sns.barplot(x = values, y = counts)

    plt.title(a)
    plt.xticks(rotation=45)
    print(a, values, counts)
```

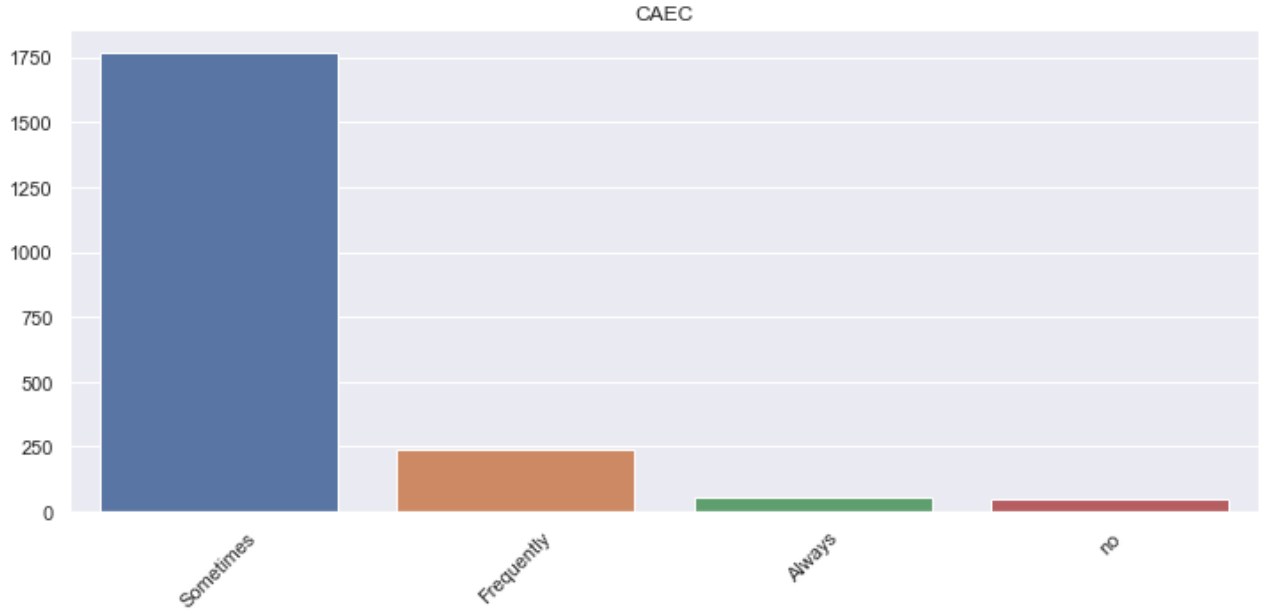
```
family_history_with_overweight ['yes', 'no'] [1726, 385]
FAVC ['yes', 'no'] [1866, 245]
FCVC ['Sometimes', 'Always', 'Never'] [1013, 996, 102]
NCP ['3', '1', '2', '3+'] [1470, 316, 176, 149]
CAEC ['Sometimes', 'Frequently', 'Always', 'no'] [1765, 242, 53, 51]
SMOKE ['no', 'yes'] [2067, 44]
CH2O ['Between 1 and 2 L', 'More than 2 L', 'Less than a liter'] [1110, 516, 48
5]
SCC ['no', 'yes'] [2015, 96]
FAF ['1 or 2 days', 'I do not have', '2 or 4 days', '4 or 5 days'] [776, 720, 49
6, 119]
TUE ['0–2 hours', '3–5 hours', 'More than 5 hours'] [952, 915, 244]
CALC ['Sometimes', 'no', 'Frequently', 'Always'] [1401, 639, 70, 1]
MTRANS ['Public Transportation', 'Automobile', 'Walking', 'Motorbike', 'Bike']
[1580, 457, 56, 11, 7]
```

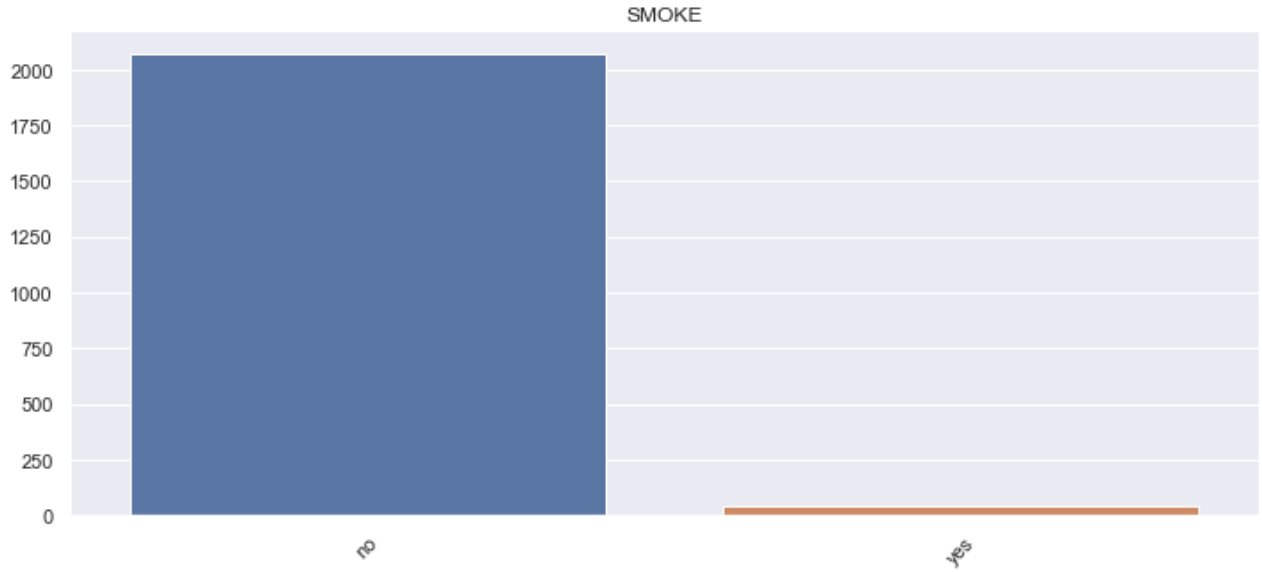

family_history_with_overweight
1750
1500
1250
1000
750
500
250
0
yes
no

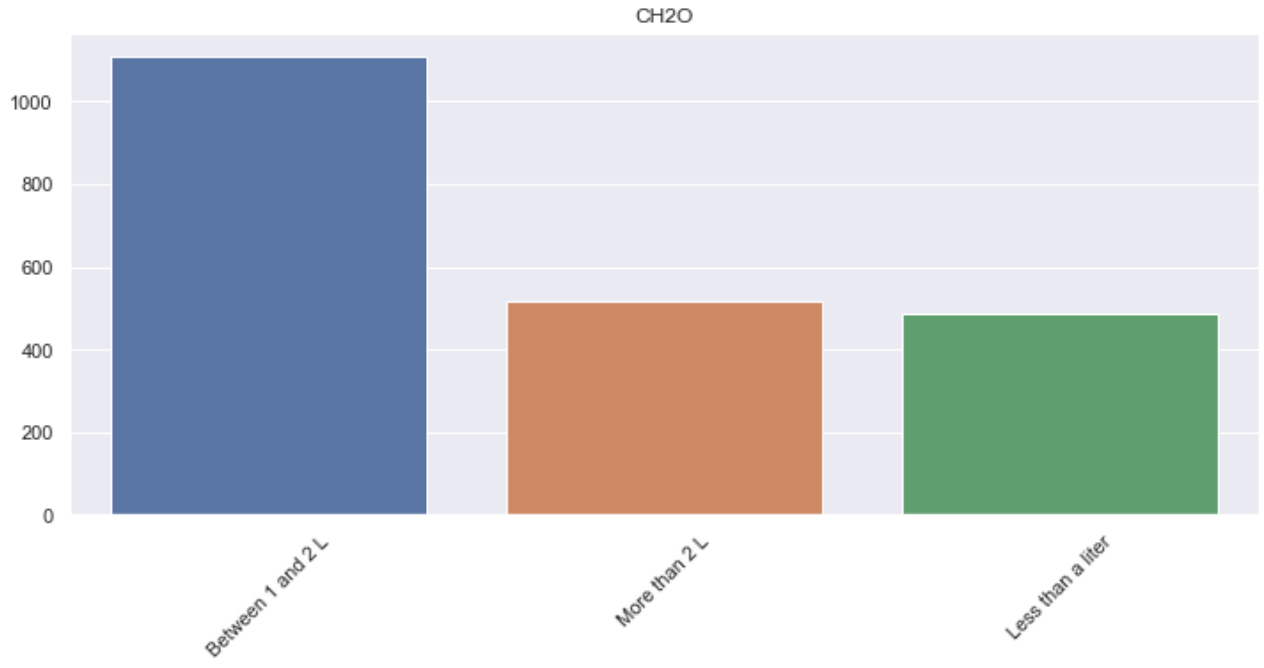

FAVC
1750
1500
1250
1000
750
500
250
0
yes
no

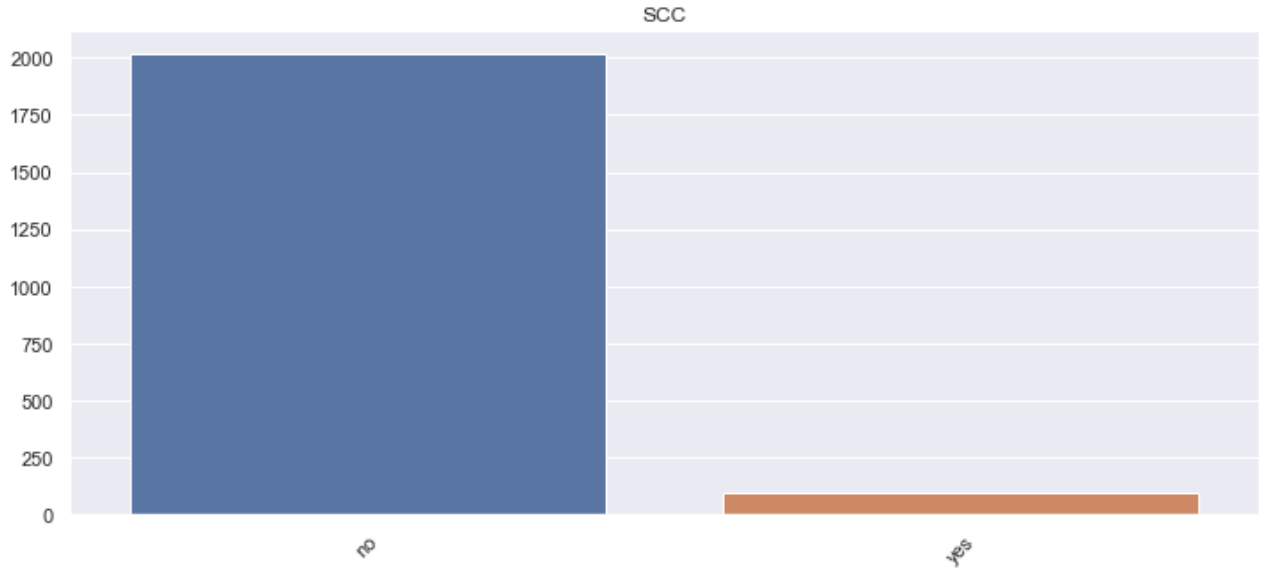

FCVC
1000
800
600
400
200
0
Sometimes
Always
Never

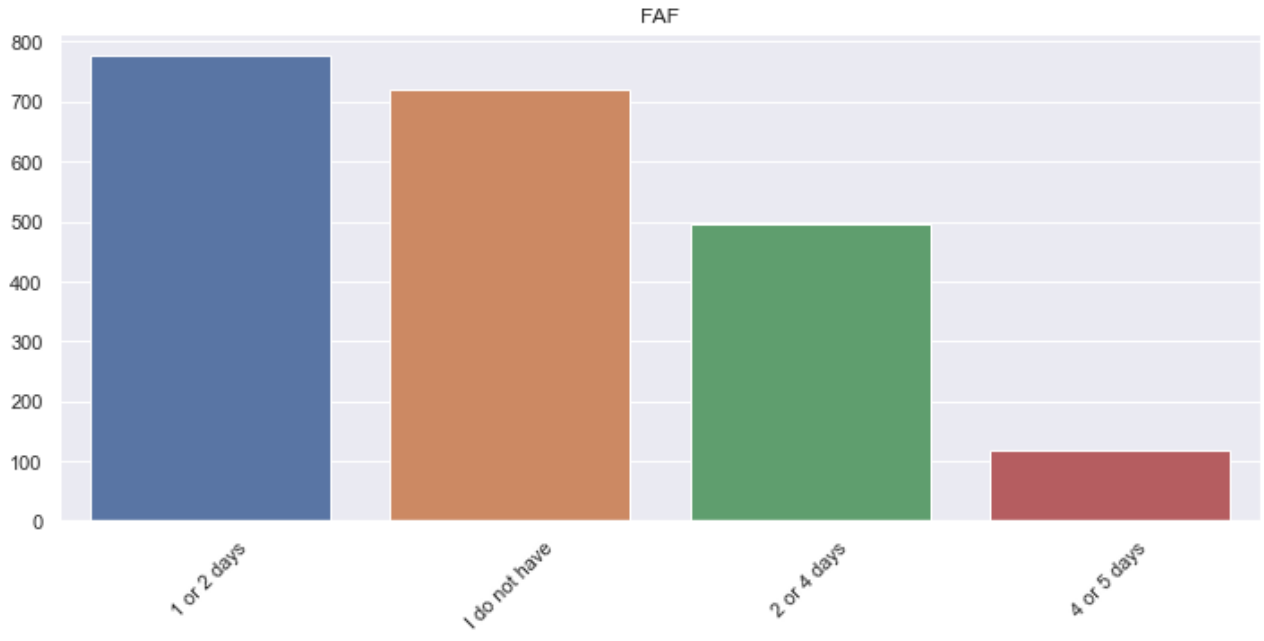

NCP
1400
1200
1000
800
600
400
200
0
3
1
2
3+

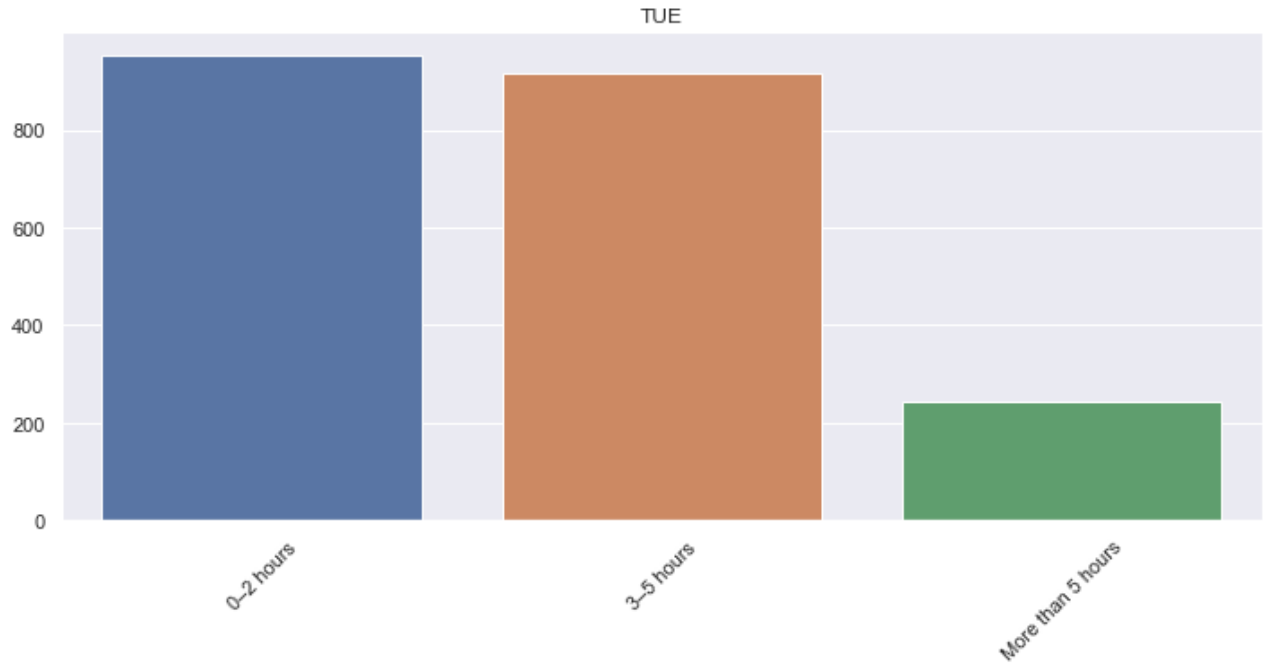

CAEC
1750
1500
1250
1000
750
500
250
0
Sometimes
Frequently
Always
no

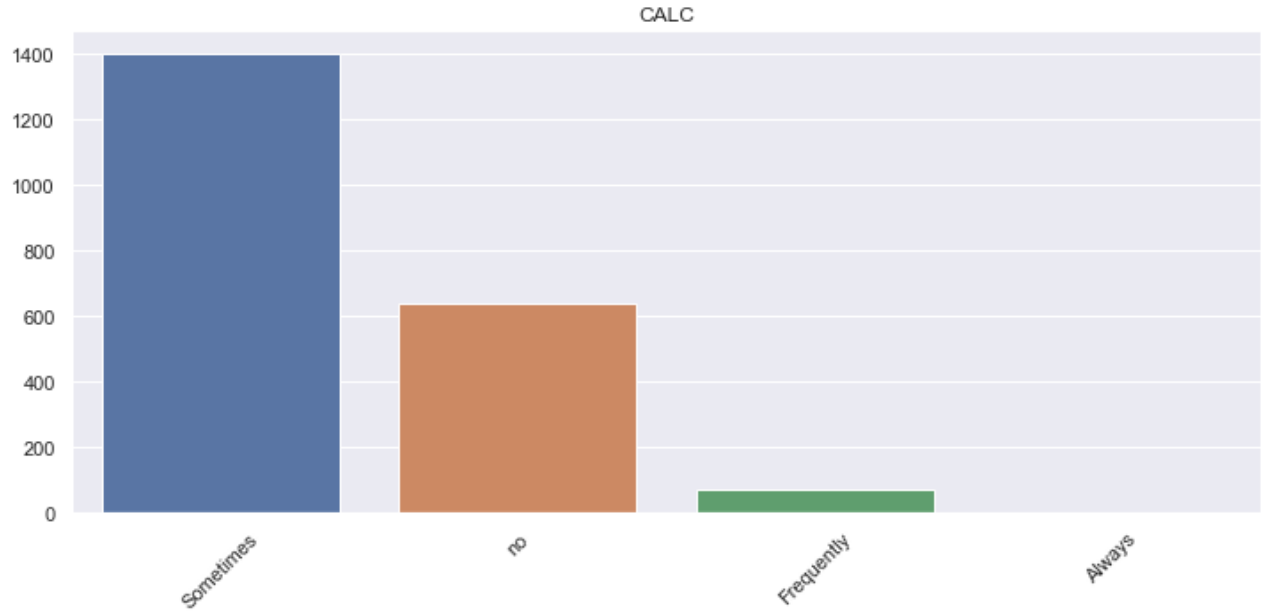

SMOKE
2000
1750
1500
1250
1000
750
500
250
0
no
yes

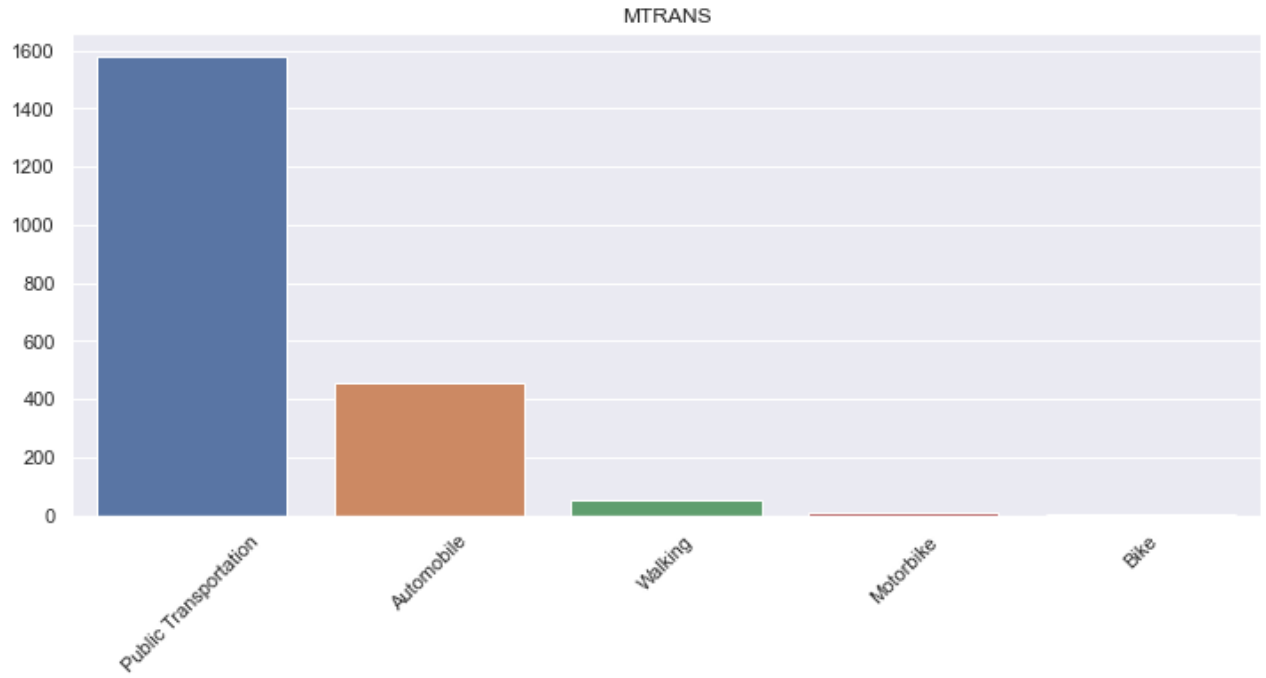

CH2O
1000
800
600
400
200
0
Between 1 and 2 L
More than 2 L
Less than a liter


SCC
2000
1750
1500
1250
1000
750
500
250
0
no
yes


FAF
800
700
600
500
400
300
200
100
0
1 or 2 days
I do not have
2 or 4 days
4 or 5 days


TUE
800
600
400
200
0
0–2 hours
3–5 hours
More than 5 hours


CALC
1400
1200
1000
800
600
400
200
0
Sometimes
no
Frequently
Always



Data Preprossing

In [30]:
```
df1.head()
```

Out[30]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC
0	Female	21.0	162.0	64.0	yes	no	2	3	Sometimes
1	Female	21.0	152.0	56.0	yes	no	3	3	Sometimes
2	Male	23.0	180.0	77.0	yes	no	2	3	Sometimes
3	Male	27.0	180.0	87.0	no	no	3	3	Sometimes
4	Male	22.0	178.0	89.8	no	no	2	1	Sometimes

Since classifier cannot operate with label data directly, One Hot Encoder and Label Encoding will be used to assign numeric values to each category

In [31]:
```
# identity categorical variables (data type would be 'object')
cat = df1.dtypes == object

print(cat)

# When dtype == object is 'true'
print(cat[cat])
cat_labels = cat[cat].index
print('Categorical variables:', cat_labels)

# When dtype == object is 'false'
false = cat[~cat]
```

```
non_cat = false.index
print('Non Categorical variables:', non_cat)
```

```
Gender                                True
Age                                  False
Height                               False
Weight                               False
family_history_with_overweight        True
FAVC                                  True
FCVC                                 False
NCP                                  False
CAEC                                  True
SMOKE                                 True
CH2O                                 False
SCC                                   True
FAF                                  False
TUE                                  False
CALC                                  True
MTRANS                                True
NObeyesdad                            True
dtype: bool
Gender                               True
family_history_with_overweight       True
FAVC                                 True
CAEC                                 True
SMOKE                                True
SCC                                  True
CALC                                 True
MTRANS                               True
NObeyesdad                           True
dtype: bool
Categorical variables: Index(['Gender', 'family_history_with_overweight', 'FAV
C', 'CAEC', 'SMOKE',
       'SCC', 'CALC', 'MTRANS', 'NObeyesdad'],
      dtype='object')
Non Categorical variables: Index(['Age', 'Height', 'Weight', 'FCVC', 'NCP', 'CH2
O', 'FAF', 'TUE'], dtype='object')
```

In [35]:

```
df1.head(3)
```

Out[35]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC
0	Female	21.0	162.0	64.0	yes	no	2	3	Sometimes
1	Female	21.0	152.0	56.0	yes	no	3	3	Sometimes
2	Male	23.0	180.0	77.0	yes	no	2	3	Sometimes

In [36]:

```
df1.columns

def col_no(x):
    d = {}
    d[df1.columns[x]] = x
    return(d)

print([col_no(x) for x in range(0, len(df1.columns))])
```

```
[{'Gender': 0}, {'Age': 1}, {'Height': 2}, {'Weight': 3}, {'family_history_with_
```

```
overweight': 4}, {'FAVC': 5}, {'FCVC': 6}, {'NCP': 7}, {'CAEC': 8}, {'SMOKE':
9}, {'CH2O': 10}, {'SCC': 11}, {'FAF': 12}, {'TUE': 13}, {'CALC': 14}, {'MTRAN
S': 15}, {'NObeyesdad': 16}]
```

In [37]:
```
x = df1[df1.columns[:-1]]
y = df['NObeyesdad']

x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(x, y
```

The target value, obesity level, will be transformed into digit label with LabelEncoder.

StandardScaler is applied to attributes with values which ranges are not consistent with the rest, to avoid disproportionate weight assigned to these values. (i.e. Age, Height, Weight).

Features that are ordinal in nature (i.e. answers including 'never', 'sometimes', 'always') will be preprocessed with OrdinalEncoder (exactly the same function is LabelEncoder, however this will take in multiple arguments as the latter is meant for the y-value only).

Features that are non-ordinal in nature will be preprocessed with OneHotEncoder, so that the generated labels will not be interpreted in a way that suggests one answer is more important than the other (e.g. 3 is more important than 1).

SimpleImputer is applied to all attributes to deal with missing values.

All of these preprocessing techniques will be bundled into a pipeline, which will be deployed with classifiers later.

In [39]:
```
from sklearn.preprocessing import LabelEncoder
```

In [40]:
```
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_train
```

Out[40]:
```
array([2, 4, 2, ..., 3, 1, 5])
```

In [44]:
```
Scale_features = ['Age', 'Height', 'Weight']
Scale_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('Scaling', StandardScaler())
])

Ordi_features = ['CAEC', 'CALC']
Ordi_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('Ordi', OrdinalEncoder())
])

NonO_features = ['Gender', 'family_history_with_overweight', 'FAVC', 'SMOKE', 'S
NonO_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('Non-O', OneHotEncoder())
])

Preprocessor = ColumnTransformer(transformers=[
    ('Scale', Scale_transformer, Scale_features),
    ('Ordinal', Ordi_transformer, Ordi_features),
    ('Non-Ordinal', NonO_transformer, NonO_features)
```

```
], remainder = 'passthrough')

clf = Pipeline(steps=[('preprocessor', Preprocessor)])
```

In [45]:
```
clf.fit(x_train, y_train)
```

Out[45]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('Scale',
                                                  Pipeline(steps=[('imputer',
                                                                   SimpleImputer
(strategy='median')),
                                                                  ('Scaling',
                                                                   StandardScale
r())]),
                                                  ['Age', 'Height', 'Weight']),
                                                 ('Ordinal',
                                                  Pipeline(steps=[('imputer',
                                                                   SimpleImputer
(fill_value='missing',

strategy='constant')),
                                                                  ('Ordi',
                                                                   OrdinalEncode
r())]),
                                                  ['CAEC', 'CALC']),
                                                 ('Non-Ordinal',
                                                  Pipeline(steps=[('imputer',
                                                                   SimpleImputer
(fill_value='missing',

strategy='constant')),
                                                                  ('Non-O',
                                                                   OneHotEncoder
())]),
                                                  ['Gender',
                                                   'family_history_with_overweig
ht',

                                                   'FAVC', 'SMOKE', 'SCC',
                                                   'MTRANS'])]))])
```

In [46]:
```
trans_df = clf.fit_transform(x_train)
print(trans_df.shape)
```

```
(1899, 25)
```

In [47]:
```
# Column name of first two steps in pipeline

cols = [y for x in [Scale_features, Ordi_features] for y in x]
cols
```

Out[47]:
```
['Age', 'Height', 'Weight', 'CAEC', 'CALC']
```

In [48]:
```
# Column names of OneHotEncoder step in pipeline

ohe_cols = clf.named_steps['preprocessor'].transformers_[2][1]\
    .named_steps['Non-O'].get_feature_names(NonO_features)
```

```
ohe_cols = [x for x in ohe_cols]
ohe_cols
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/deprecation.py:87: Futu
reWarning: Function get_feature_names is deprecated; get_feature_names is deprec
ated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instea
d.
  warnings.warn(msg, category=FutureWarning)
```

Out[48]:
```
['Gender_Female',
 'Gender_Male',
 'family_history_with_overweight_no',
 'family_history_with_overweight_yes',
 'FAVC_no',
 'FAVC_yes',
 'SMOKE_no',
 'SMOKE_yes',
 'SCC_no',
 'SCC_yes',
 'MTRANS_Automobile',
 'MTRANS_Bike',
 'MTRANS_Motorbike',
 'MTRANS_Public Transportation',
 'MTRANS_Walking']
```

In [49]:
```
# Column names of remainder='Passthrough' - remaining columns that didn't get pr
non_cat
```

Out[49]:
```
Index(['Age', 'Height', 'Weight', 'FCVC', 'NCP', 'CH2O', 'FAF', 'TUE'], dtype='o
bject')
```

In [50]:
```
transformed_x_train = pd.DataFrame(trans_df, columns= ['Age', 'Height',
 'Weight',
 'CAEC', 'CALC','Gender_Female',
 'Gender_Male',
 'Family History with Overweight_no',
 'Family History with Overweight_yes',
 'FAVC_no',
 'FAVC_yes',
 'SMOKE_no',
 'SMOKE_yes',
 'SCC_no',
 'SCC_yes',
 'MTRANS_Automobile',
 'MTRANS_Bike',
 'MTRANS_Motorbike',
 'MTRANS_Public Transportation',
 'MTRANS_Walking',
 'FCVC', 'NCP', 'CH2O', 'FAF', 'TUE'])
```

In [51]:
```
# transformed/processed features

transformed_x_train
```

Out[51]:

	Age	Height	Weight	CAEC	CALC	Gender_Female	Gender_Male	Family Histo wi Overweight_
0	0.265303	-0.467044	0.128819	2.0	2.0	0.0	1.0	0

	Age	Height	Weight	CAEC	CALC	Gender_Female	Gender_Male	Family Histo wi Overweight_
1	0.265303	-1.012489	0.697446	2.0	2.0	1.0	0.0	C
2	1.346780	0.891223	0.651956	2.0	2.0	0.0	1.0	C
3	-0.831847	0.623847	1.793002	2.0	2.0	1.0	0.0	C
4	-0.769152	-0.777199	-0.174449	2.0	2.0	1.0	0.0	C
...	...	...	...	...	...	...	...	
1894	0.641469	0.709407	1.008296	2.0	2.0	0.0	1.0	C
1895	-0.675111	-1.076659	-1.387521	2.0	3.0	1.0	0.0	'
1896	0.986288	0.709407	1.205420	2.0	2.0	0.0	1.0	C
1897	0.108568	-1.397509	-1.197978	2.0	2.0	1.0	0.0	'
1898	0.422039	-1.611409	-0.932619	2.0	2.0	1.0	0.0	'

1899 rows × 25 columns

In [52]:
```
le = LabelEncoder()
y_test = le.fit_transform(y_test)
le_name_mapping = dict(zip(le.transform(le.classes_), le.classes_))
print(le_name_mapping)
```

```
{0: 'Insufficient Weight', 1: 'Normal Weight', 2: 'Obesity Type I', 3: 'Obesity
Type II', 4: 'Obesity Type III', 5: 'Overweight Level I', 6: 'Overweight Level I
I'}
```

Appendix A2

Model Selection

In [54]:
```
classifiers = [
    KNeighborsClassifier(n_neighbors = 5),
    SVC(kernel="rbf", C=0.025, probability=True),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    AdaBoostClassifier(),
    GradientBoostingClassifier(),
    SGDClassifier()
    ]

top_class = []

for classifier in classifiers:
    pipe = Pipeline(steps=[('preprocessor', Preprocessor),
                           ('classifier', classifier)])

    # training model
    pipe.fit(x_train, y_train)
    print(classifier)
```

```
    acc_score = pipe.score(x_test, y_test)
    print("model score: %.3f" % acc_score)

    # using the model to predict
    y_pred = pipe.predict(x_test)

    target_names = [le_name_mapping[x] for x in le_name_mapping]
    print(classification_report(y_test, y_pred, target_names=target_names))

    if acc_score > 0.8:
        top_class.append(classifier)
```

```
KNeighborsClassifier()
model score: 0.821
                     precision    recall  f1-score   support

Insufficient Weight       0.72      0.92      0.81        25
      Normal Weight       0.61      0.35      0.45        31
     Obesity Type I       0.84      0.95      0.89        44
    Obesity Type II       0.94      1.00      0.97        31
   Obesity Type III       1.00      1.00      1.00        27
 Overweight Level I       0.77      0.82      0.79        28
Overweight Level II       0.77      0.65      0.71        26

           accuracy                           0.82       212
          macro avg       0.81      0.81      0.80       212
       weighted avg       0.81      0.82      0.81       212

SVC(C=0.025, probability=True)
model score: 0.505
                     precision    recall  f1-score   support

Insufficient Weight       0.67      0.24      0.35        25
      Normal Weight       0.35      0.19      0.25        31
     Obesity Type I       0.34      1.00      0.51        44
    Obesity Type II       0.83      0.77      0.80        31
   Obesity Type III       1.00      1.00      1.00        27
 Overweight Level I       0.00      0.00      0.00        28
Overweight Level II       0.00      0.00      0.00        26

           accuracy                           0.50       212
          macro avg       0.46      0.46      0.42       212
       weighted avg       0.45      0.50      0.43       212

DecisionTreeClassifier()
model score: 0.939
                     precision    recall  f1-score   support

Insufficient Weight       0.96      0.92      0.94        25
      Normal Weight       0.90      0.87      0.89        31
     Obesity Type I       0.95      0.95      0.95        44
    Obesity Type II       0.94      1.00      0.97        31
   Obesity Type III       1.00      1.00      1.00        27
 Overweight Level I       0.89      0.89      0.89        28
Overweight Level II       0.92      0.92      0.92        26

           accuracy                           0.94       212
          macro avg       0.94      0.94      0.94       212
       weighted avg       0.94      0.94      0.94       212

/opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:13
08: 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 con
```

```
trol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:13
08: 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 con
trol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:13
08: 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 con
trol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
RandomForestClassifier()
model score: 0.934
                      precision    recall  f1-score   support

Insufficient Weight       1.00      0.88      0.94        25
      Normal Weight       0.77      0.97      0.86        31
     Obesity Type I       1.00      0.95      0.98        44
    Obesity Type II       1.00      1.00      1.00        31
   Obesity Type III       1.00      1.00      1.00        27
 Overweight Level I       0.92      0.82      0.87        28
Overweight Level II       0.88      0.88      0.88        26

           accuracy                           0.93       212
          macro avg       0.94      0.93      0.93       212
       weighted avg       0.94      0.93      0.94       212

AdaBoostClassifier()
model score: 0.283
                      precision    recall  f1-score   support

Insufficient Weight       0.00      0.00      0.00        25
      Normal Weight       0.35      0.45      0.39        31
     Obesity Type I       0.25      0.43      0.31        44
    Obesity Type II       0.50      0.03      0.06        31
   Obesity Type III       0.00      0.00      0.00        27
 Overweight Level I       0.50      0.04      0.07        28
Overweight Level II       0.27      0.96      0.43        26

           accuracy                           0.28       212
          macro avg       0.27      0.27      0.18       212
       weighted avg       0.28      0.28      0.19       212

/opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:13
08: 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 con
trol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:13
08: 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 con
trol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/anaconda3/lib/python3.8/site-packages/sklearn/metrics/_classification.py:13
08: 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 con
trol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
GradientBoostingClassifier()
model score: 0.958
                      precision    recall  f1-score   support

Insufficient Weight       1.00      0.92      0.96        25
      Normal Weight       0.88      0.97      0.92        31
```

```
          Obesity Type I       0.96      1.00      0.98        44
         Obesity Type II       1.00      0.97      0.98        31
        Obesity Type III       1.00      1.00      1.00        27
      Overweight Level I       0.96      0.86      0.91        28
     Overweight Level II       0.93      0.96      0.94        26

                accuracy                           0.96       212
               macro avg       0.96      0.95      0.96       212
            weighted avg       0.96      0.96      0.96       212

SGDClassifier()
model score: 0.575
                     precision    recall  f1-score   support

Insufficient Weight       1.00      0.60      0.75        25
      Normal Weight       0.30      0.94      0.45        31
     Obesity Type I       0.92      0.25      0.39        44
    Obesity Type II       0.82      1.00      0.90        31
   Obesity Type III       1.00      1.00      1.00        27
 Overweight Level I       0.37      0.25      0.30        28
Overweight Level II       0.50      0.08      0.13        26

           accuracy                           0.58       212
          macro avg       0.70      0.59      0.56       212
       weighted avg       0.71      0.58      0.55       212
```

Classification Report

In [55]:
```
top_class
```

Out[55]:
```
[KNeighborsClassifier(),
 DecisionTreeClassifier(),
 RandomForestClassifier(),
 GradientBoostingClassifier()]
```

In [ ]:
```

```

Appendix B: Preprocessing and Clustering Analysis Exploration

Cluster analysis using K-Means algorithm is performed on the full dataset. To perform the cluster analysis, first, the class label, NObeyesdad is removed from the dataset. The dataset is transformed to ensure the correct data types exist for each feature. Dummy variables are created for the categorical features. The numeric dataset contains 2,111 rows and 43 columns.

In [1]:
```
import numpy as np
import pylab as pl
import pandas as pd
import importlib
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import decomposition
from sklearn import preprocessing
from sklearn import metrics
from sklearn.metrics import completeness_score, homogeneity_score
from sklearn.metrics import silhouette_samples
```

In [2]:
```
%pwd
```

Out[2]: '/Users/cl'

In [3]:
```
# Load dataset to Pandas dataframe:
df = pd.read_csv('/Users/cl/ObesityDataset.csv', header=0)
```

In [4]:
```
# View dataframe:
df
```

Out[4]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	N
0	Female	21.000000	1.620000	64.000000	yes	no	2.0	
1	Female	21.000000	1.520000	56.000000	yes	no	3.0	
2	Male	23.000000	1.800000	77.000000	yes	no	2.0	
3	Male	27.000000	1.800000	87.000000	no	no	3.0	
4	Male	22.000000	1.780000	89.800000	no	no	2.0	
...	...	...	...	...	...	...	...	
2106	Female	20.976842	1.710730	131.408528	yes	yes	3.0	
2107	Female	21.982942	1.748584	133.742943	yes	yes	3.0	
2108	Female	22.524036	1.752206	133.689352	yes	yes	3.0	
2109	Female	24.361936	1.739450	133.346641	yes	yes	3.0	
2110	Female	23.664709	1.738836	133.472641	yes	yes	3.0	

2111 rows × 17 columns

In [5]:
```
#remove the class label column
df2 = df.iloc[:,:16]
df2
```

Out[5]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	N
0	Female	21.000000	1.620000	64.000000	yes	no	2.0	
1	Female	21.000000	1.520000	56.000000	yes	no	3.0	
2	Male	23.000000	1.800000	77.000000	yes	no	2.0	
3	Male	27.000000	1.800000	87.000000	no	no	3.0	
4	Male	22.000000	1.780000	89.800000	no	no	2.0	
...	...	...	...	...	...	...	...	
2106	Female	20.976842	1.710730	131.408528	yes	yes	3.0	
2107	Female	21.982942	1.748584	133.742943	yes	yes	3.0	
2108	Female	22.524036	1.752206	133.689352	yes	yes	3.0	
2109	Female	24.361936	1.739450	133.346641	yes	yes	3.0	
2110	Female	23.664709	1.738836	133.472641	yes	yes	3.0	

2111 rows × 16 columns

In [6]:
```
# Create a copy to clean the data:
cleaned_data = df2
cleaned_data
```

Out[6]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	N
0	Female	21.000000	1.620000	64.000000	yes	no	2.0	
1	Female	21.000000	1.520000	56.000000	yes	no	3.0	
2	Male	23.000000	1.800000	77.000000	yes	no	2.0	
3	Male	27.000000	1.800000	87.000000	no	no	3.0	
4	Male	22.000000	1.780000	89.800000	no	no	2.0	
...	...	...	...	...	...	...	...	
2106	Female	20.976842	1.710730	131.408528	yes	yes	3.0	
2107	Female	21.982942	1.748584	133.742943	yes	yes	3.0	
2108	Female	22.524036	1.752206	133.689352	yes	yes	3.0	
2109	Female	24.361936	1.739450	133.346641	yes	yes	3.0	
2110	Female	23.664709	1.738836	133.472641	yes	yes	3.0	

2111 rows × 16 columns

In [7]:
```
# Convert FCVC, NCP, CH20, FAF, and TUE into a Categorical Feature by first, con
cleaned_data['FCVC'] = cleaned_data['FCVC'].astype('int')
cleaned_data['NCP'] = cleaned_data['NCP'].astype('int')
```

```
cleaned_data['CH2O'] = cleaned_data['CH2O'].astype('int')
cleaned_data['FAF'] = cleaned_data['FAF'].astype('int')
cleaned_data['TUE'] = cleaned_data['TUE'].astype('int')

# Convert Age from Float to Integer:
cleaned_data['Age'] = cleaned_data['Age'].astype('int')
cleaned_data.dtypes
```

Out[7]:
```
Gender                            object
Age                                int64
Height                           float64
Weight                           float64
family_history_with_overweight    object
FAVC                              object
FCVC                               int64
NCP                                int64
CAEC                              object
SMOKE                             object
CH2O                               int64
SCC                               object
FAF                                int64
TUE                                int64
CALC                              object
MTRANS                            object
dtype: object
```

In [8]:
```
# Rename values in FCVC into Categorical Names:
cleaned_data['FCVC'] = cleaned_data['FCVC'].replace({1: 'Never'})
cleaned_data['FCVC'] = cleaned_data['FCVC'].replace({2: 'Sometimes'})
cleaned_data['FCVC'] = cleaned_data['FCVC'].replace({3: 'Always'})

# Rename values in NCP into Categorical Names:
cleaned_data['NCP'] = cleaned_data['NCP'].replace({1: '1'})
cleaned_data['NCP'] = cleaned_data['NCP'].replace({2: '2'})
cleaned_data['NCP'] = cleaned_data['NCP'].replace({3: '3'})
cleaned_data['NCP'] = cleaned_data['NCP'].replace({4: '3+'})

# Rename values in CH2O into Categorical Names:
cleaned_data['CH2O'] = cleaned_data['CH2O'].replace({1: 'Less than a liter'})
cleaned_data['CH2O'] = cleaned_data['CH2O'].replace({2: 'Between 1 and 2 L'})
cleaned_data['CH2O'] = cleaned_data['CH2O'].replace({3: 'More than 2 L'})

# Rename values in FAF into Categorical Names:
cleaned_data['FAF'] = cleaned_data['FAF'].replace({0: 'I do not have'})
cleaned_data['FAF'] = cleaned_data['FAF'].replace({1: '1 or 2 days'})
cleaned_data['FAF'] = cleaned_data['FAF'].replace({2: '2 or 4 days'})
cleaned_data['FAF'] = cleaned_data['FAF'].replace({3: '4 or 5 days'})

# Rename values in TUE into Categorical Names:
cleaned_data['TUE'] = cleaned_data['TUE'].replace({0: '0-2 Hours'})
cleaned_data['TUE'] = cleaned_data['TUE'].replace({1: '3-5 Hours'})
cleaned_data['TUE'] = cleaned_data['TUE'].replace({2: 'More than 5 Hours'})

cleaned_data
```

Out[8]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCI
0	Female	21	1.620000	64.000000		yes	no	Sometimes

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCI
1	Female	21	1.520000	56.000000	yes	no	Always	
2	Male	23	1.800000	77.000000	yes	no	Sometimes	
3	Male	27	1.800000	87.000000	no	no	Always	
4	Male	22	1.780000	89.800000	no	no	Sometimes	
...	...	...	...	...	...	...	...	...
2106	Female	20	1.710730	131.408528	yes	yes	Always	
2107	Female	21	1.748584	133.742943	yes	yes	Always	
2108	Female	22	1.752206	133.689352	yes	yes	Always	
2109	Female	24	1.739450	133.346641	yes	yes	Always	
2110	Female	23	1.738836	133.472641	yes	yes	Always	

2111 rows × 16 columns

In [9]:
```
# create dummy variables for cleaned dataset:
data_numeric = pd.get_dummies(cleaned_data)
pd.set_option("display.max_columns", 999)
data_numeric
```

Out[9]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_r
0	21	1.620000	64.000000	1	0	
1	21	1.520000	56.000000	1	0	
2	23	1.800000	77.000000	0	1	
3	27	1.800000	87.000000	0	1	
4	22	1.780000	89.800000	0	1	
...	...	...	...	...	...	

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_r
2106	20	1.710730	131.408528	1	0	
2107	21	1.748584	133.742943	1	0	
2108	22	1.752206	133.689352	1	0	
2109	24	1.739450	133.346641	1	0	
2110	23	1.738836	133.472641	1	0	

2111 rows × 43 columns

In [10]:
```
# Save Numeric Dataframe for future use:
data_numeric.to_csv('/Users/cl/Obesity_numeric.csv', index = False)
```

In [11]:
```
# Normalize the numeric dataset with Min-Max Scaling:
df_min_max_scaled = data_numeric.copy()
for column in df_min_max_scaled.columns:
    df_min_max_scaled[column] = (df_min_max_scaled[column] - df_min_max_scaled[c
```

In [12]:
```
# View normalized data:
print(df_min_max_scaled)
```

```
           Age    Height    Weight  Gender_Female  Gender_Male  \
0     0.148936  0.320755  0.186567            1.0          0.0
1     0.148936  0.132075  0.126866            1.0          0.0
2     0.191489  0.660377  0.283582            0.0          1.0
3     0.276596  0.660377  0.358209            0.0          1.0
4     0.170213  0.622642  0.379104            0.0          1.0
...        ...       ...       ...            ...          ...
2106  0.127660  0.491943  0.689616            1.0          0.0
2107  0.148936  0.563366  0.707037            1.0          0.0
2108  0.170213  0.570200  0.706637            1.0          0.0
2109  0.212766  0.546132  0.704079            1.0          0.0
2110  0.191489  0.544974  0.705020            1.0          0.0

      family_history_with_overweight_no  family_history_with_overweight_yes  \
0                                   0.0                                 1.0
1                                   0.0                                 1.0
2                                   0.0                                 1.0
3                                   1.0                                 0.0
4                                   1.0                                 0.0
...                                 ...                                 ...
2106                                0.0                                 1.0
2107                                0.0                                 1.0
2108                                0.0                                 1.0
2109                                0.0                                 1.0
2110                                0.0                                 1.0

      FAVC_no  FAVC_yes  FCVC_Always  FCVC_Never  FCVC_Sometimes  NCP_1  \
0         1.0       0.0          0.0         0.0             1.0    0.0
1         1.0       0.0          1.0         0.0             0.0    0.0
2         1.0       0.0          0.0         0.0             1.0    0.0
3         1.0       0.0          1.0         0.0             0.0    0.0
```

```
4          1.0        0.0           0.0          0.0              1.0     1.0
...        ...        ...           ...          ...              ...     ...
2106       0.0        1.0           1.0          0.0              0.0     0.0
2107       0.0        1.0           1.0          0.0              0.0     0.0
2108       0.0        1.0           1.0          0.0              0.0     0.0
2109       0.0        1.0           1.0          0.0              0.0     0.0
2110       0.0        1.0           1.0          0.0              0.0     0.0

      NCP_2  NCP_3  NCP_3+  CAEC_Always  CAEC_Frequently  CAEC_Sometimes  \
0       0.0    1.0     0.0          0.0              0.0             1.0
1       0.0    1.0     0.0          0.0              0.0             1.0
2       0.0    1.0     0.0          0.0              0.0             1.0
3       0.0    1.0     0.0          0.0              0.0             1.0
4       0.0    0.0     0.0          0.0              0.0             1.0
...     ...    ...     ...          ...              ...             ...
2106    0.0    1.0     0.0          0.0              0.0             1.0
2107    0.0    1.0     0.0          0.0              0.0             1.0
2108    0.0    1.0     0.0          0.0              0.0             1.0
2109    0.0    1.0     0.0          0.0              0.0             1.0
2110    0.0    1.0     0.0          0.0              0.0             1.0

      CAEC_no  SMOKE_no  SMOKE_yes  CH2O_Between 1 and 2 L  \
0         0.0       1.0        0.0                     1.0
1         0.0       0.0        1.0                     0.0
2         0.0       1.0        0.0                     1.0
3         0.0       1.0        0.0                     1.0
4         0.0       1.0        0.0                     1.0
...       ...       ...        ...                     ...
2106      0.0       1.0        0.0                     0.0
2107      0.0       1.0        0.0                     1.0
2108      0.0       1.0        0.0                     1.0
2109      0.0       1.0        0.0                     1.0
2110      0.0       1.0        0.0                     1.0

      CH2O_Less than a liter  CH2O_More than 2 L  SCC_no  SCC_yes  \
0                        0.0                 0.0     1.0      0.0
1                        0.0                 1.0     0.0      1.0
2                        0.0                 0.0     1.0      0.0
3                        0.0                 0.0     1.0      0.0
4                        0.0                 0.0     1.0      0.0
...                      ...                 ...     ...      ...
2106                     1.0                 0.0     1.0      0.0
2107                     0.0                 0.0     1.0      0.0
2108                     0.0                 0.0     1.0      0.0
2109                     0.0                 0.0     1.0      0.0
2110                     0.0                 0.0     1.0      0.0

      FAF_1 or 2 days  FAF_2 or 4 days  FAF_4 or 5 days  FAF_I do not have  \
0                 0.0              0.0              0.0                1.0
1                 0.0              0.0              1.0                0.0
2                 0.0              1.0              0.0                0.0
3                 0.0              1.0              0.0                0.0
4                 0.0              0.0              0.0                1.0
...               ...              ...              ...                ...
2106              1.0              0.0              0.0                0.0
2107              1.0              0.0              0.0                0.0
2108              1.0              0.0              0.0                0.0
2109              1.0              0.0              0.0                0.0
2110              1.0              0.0              0.0                0.0

      TUE_0-2 Hours  TUE_3-5 Hours  TUE_More than 5 Hours  CALC_Always  \
0               0.0            1.0                    0.0          0.0
1               1.0            0.0                    0.0          0.0
2               0.0            1.0                    0.0          0.0
3               1.0            0.0                    0.0          0.0
```

```
4               1.0             0.0                     0.0           0.0
...             ...             ...                     ...           ...
2106            1.0             0.0                     0.0           0.0
2107            1.0             0.0                     0.0           0.0
2108            1.0             0.0                     0.0           0.0
2109            1.0             0.0                     0.0           0.0
2110            1.0             0.0                     0.0           0.0

      CALC_Frequently  CALC_Sometimes  CALC_no  MTRANS_Automobile  \
0                 0.0             0.0      1.0                0.0
1                 0.0             1.0      0.0                0.0
2                 1.0             0.0      0.0                0.0
3                 1.0             0.0      0.0                0.0
4                 0.0             1.0      0.0                0.0
...               ...             ...      ...                ...
2106              0.0             1.0      0.0                0.0
2107              0.0             1.0      0.0                0.0
2108              0.0             1.0      0.0                0.0
2109              0.0             1.0      0.0                0.0
2110              0.0             1.0      0.0                0.0

      MTRANS_Bike  MTRANS_Motorbike  MTRANS_Public_Transportation  \
0             0.0               0.0                           1.0
1             0.0               0.0                           1.0
2             0.0               0.0                           1.0
3             0.0               0.0                           0.0
4             0.0               0.0                           1.0
...           ...               ...                           ...
2106          0.0               0.0                           1.0
2107          0.0               0.0                           1.0
2108          0.0               0.0                           1.0
2109          0.0               0.0                           1.0
2110          0.0               0.0                           1.0

      MTRANS_Walking
0                0.0
1                0.0
2                0.0
3                1.0
4                0.0
...              ...
2106             0.0
2107             0.0
2108             0.0
2109             0.0
2110             0.0

[2111 rows x 43 columns]
```

In [13]:

```
# View class labels:
labels_df =  df['NObeyesdad']
labels_df
```

Out[13]:

```
0             Normal_Weight
1             Normal_Weight
2             Normal_Weight
3        Overweight_Level_I
4       Overweight_Level_II
              ...
2106       Obesity_Type_III
2107       Obesity_Type_III
2108       Obesity_Type_III
2109       Obesity_Type_III
```

```
2110       Obesity_Type_III
Name: NObeyesdad, Length: 2111, dtype: object
```

In [14]:
```
# Transform class label into numeric:
le = preprocessing.LabelEncoder()
labels_num = le.fit_transform(labels_df)
labels_num
```

Out[14]:
```
array([1, 1, 1, ..., 4, 4, 4])
```

In [15]:
```
# View class label names and numeric association:
label_names = dict(zip(le.transform(le.classes_), le.classes_))
print(label_names)
```

```
{0: 'Insufficient_Weight', 1: 'Normal_Weight', 2: 'Obesity_Type_I', 3: 'Obesity_
Type_II', 4: 'Obesity_Type_III', 5: 'Overweight_Level_I', 6: 'Overweight_Level_I
I'}
```

Clustering Exporation with K-Means:

Below is the exploration of clustering using K-means with the normalized data. Various values of k were tested and the centroids were evaluated to determine if a pattern appears in the clusters based on the data. For each value of K, the cluster centroids were examined to determine if any pattern exists in the data. A silhouette analysis is performed for to evaluate the separation between the resulting clusters and determine the quality of the clusters. The silhouette plots display a measure of how close each point in one cluster is to points in the neighboring clusters. The mean silhouette value is calculated and used as a threshold when determining the cluster quality. Clusters with most of their coefficients above the mean silhouette value are considered better quality which means that clusters are further away from the neighboring clusters. Clusters with most of their coefficients below the mean silhouette value reveals that samples are very close to the decision boundary between two neighboring clusters and negative coefficient values indicate that samples are assigned to the wrong cluster. When the silhouette plot does not display any negative coefficients and have the thickest plots visually above the silhouette mean, the correct number of K has been selected.

In [16]:
```
kmeans = KMeans(n_clusters=5, max_iter=500, verbose=1) #initialize k-means with
```

In [17]:
```
kmeans.fit(df_min_max_scaled)
```

```
Initialization complete
Iteration 0, inertia 11787.00508682668
Iteration 1, inertia 8566.056202078382
Iteration 2, inertia 8338.71778464269
Iteration 3, inertia 8237.82330971913
Iteration 4, inertia 8176.41527470928
Iteration 5, inertia 8099.540960517189
Iteration 6, inertia 8076.56013777138
Iteration 7, inertia 8068.9930112547
Iteration 8, inertia 8068.221972230656
Iteration 9, inertia 8068.076896983779
Iteration 10, inertia 8067.991083649056
Iteration 11, inertia 8067.917526159745
```

```
Iteration 12, inertia 8067.434112719851
Iteration 13, inertia 8067.300618383514
Iteration 14, inertia 8066.690183145394
Iteration 15, inertia 8065.724907535046
Iteration 16, inertia 8065.1555932972105
Iteration 17, inertia 8064.406578178977
Iteration 18, inertia 8055.777491215358
Iteration 19, inertia 8050.3019028424
Iteration 20, inertia 8042.536097506457
Iteration 21, inertia 8026.138983270792
Iteration 22, inertia 7994.538660965364
Iteration 23, inertia 7981.791935092773
Iteration 24, inertia 7972.431678248945
Iteration 25, inertia 7970.020550287187
Iteration 26, inertia 7968.48651338652
Iteration 27, inertia 7966.955938122063
Iteration 28, inertia 7964.781849553093
Iteration 29, inertia 7958.222148843893
Iteration 30, inertia 7956.01524580361
Iteration 31, inertia 7955.555539283562
Iteration 32, inertia 7955.527509697825
Iteration 33, inertia 7955.494519204406
Converged at iteration 33: strict convergence.
Initialization complete
Iteration 0, inertia 11825.274489539268
Iteration 1, inertia 8581.769457282187
Iteration 2, inertia 8468.439310863918
Iteration 3, inertia 8354.00074441653
Iteration 4, inertia 8210.581719370295
Iteration 5, inertia 8130.050202902669
Iteration 6, inertia 8093.4953218710225
Iteration 7, inertia 8076.467461725121
Iteration 8, inertia 8059.590914960909
Iteration 9, inertia 8051.452820847045
Iteration 10, inertia 8044.333927734175
Iteration 11, inertia 8039.893289001246
Iteration 12, inertia 8036.123631679592
Iteration 13, inertia 8032.046237708075
Iteration 14, inertia 8029.674990266088
Iteration 15, inertia 8026.254886825644
Iteration 16, inertia 8023.089343537067
Iteration 17, inertia 8019.880934083217
Iteration 18, inertia 8008.990523342801
Iteration 19, inertia 7969.300639772133
Iteration 20, inertia 7919.97712322982
Iteration 21, inertia 7910.686198322982
Iteration 22, inertia 7906.238208315535
Iteration 23, inertia 7899.756117486131
Iteration 24, inertia 7892.545244873981
Iteration 25, inertia 7890.019624999294
Iteration 26, inertia 7889.072892812861
Iteration 27, inertia 7888.575279501931
Iteration 28, inertia 7888.2990556955165
Iteration 29, inertia 7887.869135061676
Iteration 30, inertia 7887.622886266341
Iteration 31, inertia 7887.258246367743
Iteration 32, inertia 7886.838669332199
Iteration 33, inertia 7886.583695027855
Iteration 34, inertia 7886.48352229928
Converged at iteration 34: strict convergence.
Initialization complete
Iteration 0, inertia 12174.525656749454
Iteration 1, inertia 8507.8234001361
Iteration 2, inertia 8262.156761189228
Iteration 3, inertia 8135.565056552313
```

```
Iteration 4, inertia 8072.3981424163885
Iteration 5, inertia 8047.041186769761
Iteration 6, inertia 8036.157681658016
Iteration 7, inertia 8032.793289742063
Iteration 8, inertia 8030.945913788686
Iteration 9, inertia 8030.700347934296
Iteration 10, inertia 8030.595081706321
Iteration 11, inertia 8030.463150369052
Iteration 12, inertia 8030.425825530467
Converged at iteration 12: strict convergence.
Initialization complete
Iteration 0, inertia 12478.757883097682
Iteration 1, inertia 8699.700914801515
Iteration 2, inertia 8423.74786273518
Iteration 3, inertia 8246.866008200308
Iteration 4, inertia 8137.329970995708
Iteration 5, inertia 8103.804191725505
Iteration 6, inertia 8092.207693677892
Iteration 7, inertia 8089.336054818334
Iteration 8, inertia 8087.034378963835
Iteration 9, inertia 8086.246069705694
Iteration 10, inertia 8084.619362324052
Iteration 11, inertia 8082.67643373668
Iteration 12, inertia 8082.188515787069
Converged at iteration 12: strict convergence.
Initialization complete
Iteration 0, inertia 11832.53539722714
Iteration 1, inertia 8466.1801078797
Iteration 2, inertia 8226.758916757437
Iteration 3, inertia 8115.239236717611
Iteration 4, inertia 8062.949256711981
Iteration 5, inertia 8047.895976309511
Iteration 6, inertia 8035.457973021038
Iteration 7, inertia 8028.199810257129
Iteration 8, inertia 8007.65676812841
Iteration 9, inertia 7963.399143602098
Iteration 10, inertia 7936.853393483585
Iteration 11, inertia 7924.273139615089
Iteration 12, inertia 7918.766371499822
Iteration 13, inertia 7916.290733124256
Iteration 14, inertia 7915.13212820532
Iteration 15, inertia 7913.658090371352
Iteration 16, inertia 7912.216196347794
Iteration 17, inertia 7911.340690474468
Iteration 18, inertia 7911.151814669988
Iteration 19, inertia 7911.113667420209
Converged at iteration 19: strict convergence.
Initialization complete
Iteration 0, inertia 11823.752663657368
Iteration 1, inertia 8276.46780690895
Iteration 2, inertia 8098.68081942101
Iteration 3, inertia 8037.29250680346
Iteration 4, inertia 7997.19138403611
Iteration 5, inertia 7925.487303831875
Iteration 6, inertia 7872.633962217118
Iteration 7, inertia 7853.57013270058
Iteration 8, inertia 7849.995965259756
Iteration 9, inertia 7843.743550028235
Iteration 10, inertia 7837.019905347132
Iteration 11, inertia 7834.827151309026
Iteration 12, inertia 7833.546296354347
Iteration 13, inertia 7833.009378377764
Iteration 14, inertia 7832.554446255171
Iteration 15, inertia 7832.210277934463
Iteration 16, inertia 7831.1753466434475
```

```
Iteration 17, inertia 7831.074055893812
Iteration 18, inertia 7831.027292617523
Iteration 19, inertia 7830.977013027618
Converged at iteration 19: strict convergence.
Initialization complete
Iteration 0, inertia 11596.309798286276
Iteration 1, inertia 8646.1997670067
Iteration 2, inertia 8394.42223224813
Iteration 3, inertia 8250.441907007784
Iteration 4, inertia 8175.292252415538
Iteration 5, inertia 8131.1038719677545
Iteration 6, inertia 8088.55452146436
Iteration 7, inertia 8032.992455465169
Iteration 8, inertia 7978.52526813498
Iteration 9, inertia 7956.801794915094
Iteration 10, inertia 7947.744891687089
Iteration 11, inertia 7941.340182646654
Iteration 12, inertia 7937.882158960003
Iteration 13, inertia 7936.406248808135
Iteration 14, inertia 7934.818469379045
Iteration 15, inertia 7934.023804051918
Iteration 16, inertia 7932.807323423373
Iteration 17, inertia 7931.7472552303025
Iteration 18, inertia 7931.066262593041
Iteration 19, inertia 7930.419081526247
Iteration 20, inertia 7915.339857061065
Iteration 21, inertia 7886.689696466735
Iteration 22, inertia 7859.242618003011
Iteration 23, inertia 7850.823479552829
Iteration 24, inertia 7847.789675688788
Iteration 25, inertia 7846.48853874517
Iteration 26, inertia 7842.455320611688
Iteration 27, inertia 7839.4277804122285
Iteration 28, inertia 7836.555716423136
Iteration 29, inertia 7835.294209939629
Iteration 30, inertia 7834.188876497395
Iteration 31, inertia 7833.097135623066
Iteration 32, inertia 7832.671449816852
Iteration 33, inertia 7832.380999582051
Iteration 34, inertia 7832.259089755594
Iteration 35, inertia 7832.232553830457
Converged at iteration 35: strict convergence.
Initialization complete
Iteration 0, inertia 12864.885199156606
Iteration 1, inertia 8534.055041539128
Iteration 2, inertia 8208.402584930189
Iteration 3, inertia 8134.677983160359
Iteration 4, inertia 8061.954032765476
Iteration 5, inertia 8012.741078486168
Iteration 6, inertia 7973.675852821764
Iteration 7, inertia 7960.156905577035
Iteration 8, inertia 7955.948758245115
Iteration 9, inertia 7937.8692456423405
Iteration 10, inertia 7929.672981125681
Iteration 11, inertia 7912.395873582756
Iteration 12, inertia 7884.993609624193
Iteration 13, inertia 7867.601997438204
Iteration 14, inertia 7864.718177831521
Iteration 15, inertia 7863.772078183434
Iteration 16, inertia 7863.723388842099
Converged at iteration 16: strict convergence.
Initialization complete
Iteration 0, inertia 11871.688456665612
Iteration 1, inertia 8444.666140084128
Iteration 2, inertia 8172.684939816708
```

```
Iteration 3, inertia 8022.5123919112075
Iteration 4, inertia 7989.103278787667
Iteration 5, inertia 7979.526024439475
Iteration 6, inertia 7972.321268249555
Iteration 7, inertia 7965.892255679502
Iteration 8, inertia 7959.862725195303
Iteration 9, inertia 7948.061310377594
Iteration 10, inertia 7926.876581312946
Iteration 11, inertia 7922.760473428007
Iteration 12, inertia 7920.461356227191
Iteration 13, inertia 7918.936152811791
Iteration 14, inertia 7918.355463902865
Iteration 15, inertia 7918.230916735995
Converged at iteration 15: strict convergence.
Initialization complete
Iteration 0, inertia 12258.81732786365
Iteration 1, inertia 8490.528120624003
Iteration 2, inertia 8287.639034367927
Iteration 3, inertia 8199.024276953247
Iteration 4, inertia 8142.15513015745
Iteration 5, inertia 8082.207427076215
Iteration 6, inertia 8037.963976518285
Iteration 7, inertia 8007.417500726093
Iteration 8, inertia 7978.869398652809
Iteration 9, inertia 7937.0245485312225
Iteration 10, inertia 7903.678870109956
Iteration 11, inertia 7866.78478195413
Iteration 12, inertia 7853.215930528641
Iteration 13, inertia 7847.326673615622
Iteration 14, inertia 7845.271749303697
Iteration 15, inertia 7844.525551419671
Iteration 16, inertia 7844.433188252878
Iteration 17, inertia 7844.348811651154
Converged at iteration 17: strict convergence.
```

Out[17]:
```
KMeans(max_iter=500, n_clusters=5, verbose=1)
```

In [18]:
```
clusters5 = kmeans.predict(df_min_max_scaled)
```

In [19]:
```
pd.DataFrame(clusters5, columns=["Cluster"])
```

Out[19]:

	Cluster
0	2
1	0
2	2
3	3
4	3
...	...
2106	0
2107	0
2108	0
2109	0
2110	0

2111 rows × 1 columns

In [20]:
```
def cluster_sizes(clusters):
    #clusters is an array of cluster labels for each instance in the data

    size = {}
    cluster_labels = np.unique(clusters)
    n_clusters = cluster_labels.shape[0]

    for c in cluster_labels:
        size[c] = len(df[clusters == c])
    return size
```

In [21]:
```
size5 = cluster_sizes(clusters5)

for c5 in size5.keys():
    print("Size of Cluster", c5, "= ", size5[c5])
```

```
Size of Cluster 0 =  420
Size of Cluster 1 =  423
Size of Cluster 2 =  455
Size of Cluster 3 =  355
Size of Cluster 4 =  458
```

In [22]:
```
# The centroids provide an aggregate representation and a characterization of ea
pd.options.display.float_format='{:,.2f}'.format

centroids5 = pd.DataFrame(kmeans.cluster_centers_, columns=df_min_max_scaled.col
centroids5
```

Out[22]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	family
0	0.19	0.43	0.52	1.00	0.00	0.00	
1	0.37	0.51	0.36	0.33	0.67	0.06	
2	0.16	0.41	0.30	0.55	0.45	0.00	
3	0.15	0.37	0.14	0.65	0.35	0.98	
4	0.18	0.63	0.42	0.00	1.00	0.02	

In [23]:
```
# Silhouette Analysis at n = 5:
c5_silhouette = metrics.silhouette_samples(df_min_max_scaled, clusters5)
print('Mean Silhouette Value :', c5_silhouette.mean())
```

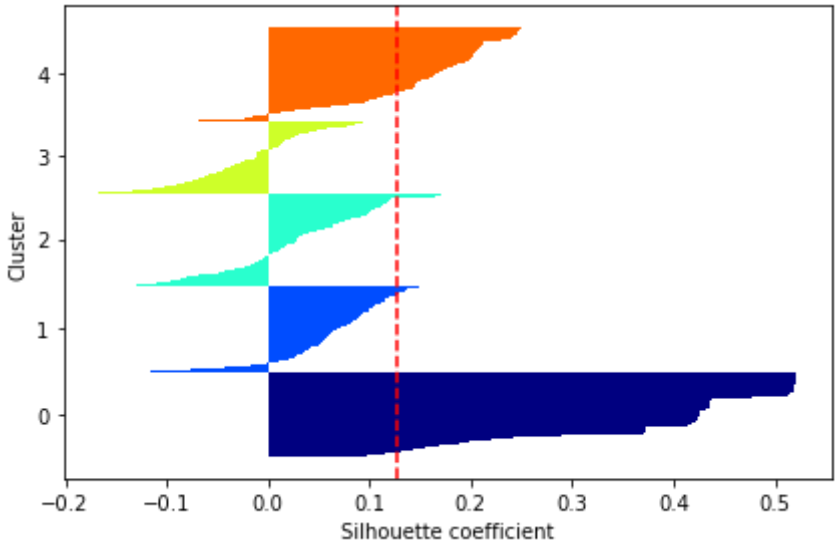
```
Mean Silhouette Value : 0.12696817428675627
```

In [24]:
```
def plot_silhouettes(data, clusters, metric='euclidean'):

    from matplotlib import cm
    from sklearn.metrics import silhouette_samples
```

```
    cluster_labels = np.unique(clusters)
    n_clusters = cluster_labels.shape[0]
    silhouette_vals = metrics.silhouette_samples(data, clusters, metric='euclide
    c_ax_lower, c_ax_upper = 0, 0
    cticks = []
    for i, k in enumerate(cluster_labels):
        c_silhouette_vals = silhouette_vals[clusters == k]
        c_silhouette_vals.sort()
        c_ax_upper += len(c_silhouette_vals)
        color = cm.jet(float(i) / n_clusters)
        pl.barh(range(c_ax_lower, c_ax_upper), c_silhouette_vals, height=1.0,
                edgecolor='none', color=color)

        cticks.append((c_ax_lower + c_ax_upper) / 2)
        c_ax_lower += len(c_silhouette_vals)

    silhouette_avg = np.mean(silhouette_vals)
    pl.axvline(silhouette_avg, color="red", linestyle="--")

    pl.yticks(cticks, cluster_labels)
    pl.ylabel('Cluster')
    pl.xlabel('Silhouette coefficient')

    pl.tight_layout()
    #pl.savefig('images/11_04.png', dpi=300)
    pl.show()

    return
```

In [25]:

```
# Plot and Evaluate the Silhouettes:
plot_silhouettes(df_min_max_scaled, clusters5)
```



Above, the plot of the silhouettes shows that cluster 0 outperformed the other clusters with all its coefficients above the mean silhouette value. Cluster 4 also performed well with many of its coefficients above the mean silhouette value. The remaining three clusters did not perform as well since most of their coefficients are below the mean silhouette value. Four of the clusters display negative values with cluster 3 having the most negative coefficients, which indicates that 5 clusters are too high for the dataset.

In [26]:
```
kmeans3 = KMeans(n_clusters=3, max_iter=500, verbose=1) # k-means with n = 3
```

In [27]:
```
kmeans3.fit(df_min_max_scaled)
```

```
Initialization complete
Iteration 0, inertia 12879.623786128155
Iteration 1, inertia 9001.37311540421
Iteration 2, inertia 8858.998139520381
Iteration 3, inertia 8809.939108030303
Iteration 4, inertia 8791.272204868248
Iteration 5, inertia 8786.47670789417
Iteration 6, inertia 8785.4661617069
Iteration 7, inertia 8783.363746212328
Iteration 8, inertia 8772.54056168085
Iteration 9, inertia 8769.557983795672
Iteration 10, inertia 8768.483498001466
Iteration 11, inertia 8767.764124919666
Iteration 12, inertia 8766.35120571748
Iteration 13, inertia 8765.43867230093
Iteration 14, inertia 8765.398780834916
Converged at iteration 14: strict convergence.
Initialization complete
Iteration 0, inertia 14534.508672109176
Iteration 1, inertia 9309.441488842096
Iteration 2, inertia 9037.604181226367
Iteration 3, inertia 8938.658231693586
Iteration 4, inertia 8916.402643453937
Iteration 5, inertia 8888.331390013225
Iteration 6, inertia 8870.074074790271
Iteration 7, inertia 8861.211103290092
Iteration 8, inertia 8853.162119584285
Iteration 9, inertia 8846.868576454352
Iteration 10, inertia 8827.84407699484
Iteration 11, inertia 8799.1609794804
Iteration 12, inertia 8785.909269034124
Iteration 13, inertia 8774.601325517255
Iteration 14, inertia 8769.592571719972
Iteration 15, inertia 8767.63492356255
Iteration 16, inertia 8766.444849568958
Iteration 17, inertia 8765.971992499608
Iteration 18, inertia 8765.867441373892
Iteration 19, inertia 8765.66116613612
Iteration 20, inertia 8765.486144212005
Iteration 21, inertia 8765.461045034992
Converged at iteration 21: strict convergence.
Initialization complete
Iteration 0, inertia 13336.164818796795
Iteration 1, inertia 9567.269152448953
Iteration 2, inertia 9445.553561777502
Iteration 3, inertia 9356.780028542624
Iteration 4, inertia 9157.130002021417
Iteration 5, inertia 9032.754433029859
Iteration 6, inertia 8966.21740202369
Iteration 7, inertia 8938.469471906204
Iteration 8, inertia 8923.340927558393
Iteration 9, inertia 8916.997734419223
Iteration 10, inertia 8915.764064629098
Iteration 11, inertia 8915.37813313638
Iteration 12, inertia 8915.127678549246
Iteration 13, inertia 8914.954033413907
Iteration 14, inertia 8914.93152766286
Iteration 15, inertia 8914.917473861125
Converged at iteration 15: strict convergence.
```

```
Initialization complete
Iteration 0, inertia 12235.950953731766
Iteration 1, inertia 9177.059040690088
Iteration 2, inertia 8997.9480370336
Iteration 3, inertia 8886.478645540206
Iteration 4, inertia 8833.780310727954
Iteration 5, inertia 8820.17237743324
Iteration 6, inertia 8819.429241417094
Iteration 7, inertia 8819.16005670056
Iteration 8, inertia 8818.804519317317
Iteration 9, inertia 8818.682843171513
Iteration 10, inertia 8818.311487274063
Iteration 11, inertia 8818.159940170133
Iteration 12, inertia 8817.98907376872
Iteration 13, inertia 8817.865936421687
Iteration 14, inertia 8817.806917535885
Iteration 15, inertia 8817.776092265041
Iteration 16, inertia 8817.693711678381
Iteration 17, inertia 8817.668602183467
Converged at iteration 17: strict convergence.
Initialization complete
Iteration 0, inertia 14283.23672461032
Iteration 1, inertia 9655.242214042786
Iteration 2, inertia 9501.150166979987
Iteration 3, inertia 9385.8241565813
Iteration 4, inertia 9306.25122999164
Iteration 5, inertia 9258.611440798086
Iteration 6, inertia 9220.655830148746
Iteration 7, inertia 9205.884675342035
Iteration 8, inertia 9201.291862951928
Iteration 9, inertia 9200.842008469204
Iteration 10, inertia 9200.67414699048
Iteration 11, inertia 9200.629965113641
Converged at iteration 11: strict convergence.
Initialization complete
Iteration 0, inertia 12423.535126370873
Iteration 1, inertia 9135.382864146819
Iteration 2, inertia 9028.876396630194
Iteration 3, inertia 8968.812555470771
Iteration 4, inertia 8910.48089075226
Iteration 5, inertia 8856.621151522155
Iteration 6, inertia 8821.480910675746
Iteration 7, inertia 8818.661138265279
Iteration 8, inertia 8818.274441804033
Iteration 9, inertia 8818.095197547456
Iteration 10, inertia 8817.924137895994
Iteration 11, inertia 8817.828887772916
Iteration 12, inertia 8817.806917535885
Iteration 13, inertia 8817.776092265041
Iteration 14, inertia 8817.693711678381
Iteration 15, inertia 8817.668602183467
Converged at iteration 15: strict convergence.
Initialization complete
Iteration 0, inertia 15560.349375046955
Iteration 1, inertia 9451.504598782774
Iteration 2, inertia 9142.603376735178
Iteration 3, inertia 8984.738489060503
Iteration 4, inertia 8903.206814994957
Iteration 5, inertia 8849.255938606113
Iteration 6, inertia 8805.887151861752
Iteration 7, inertia 8787.427510335308
Iteration 8, inertia 8777.334874885435
Iteration 9, inertia 8773.33734831006
Iteration 10, inertia 8770.052840575934
Iteration 11, inertia 8768.924265588485
```

```
Iteration 12, inertia 8768.062375098845
Iteration 13, inertia 8766.037804380696
Iteration 14, inertia 8765.5613568244
Iteration 15, inertia 8765.461045034992
Converged at iteration 15: strict convergence.
Initialization complete
Iteration 0, inertia 14256.23978990368
Iteration 1, inertia 9202.001056480733
Iteration 2, inertia 8958.453465999479
Iteration 3, inertia 8866.570788484862
Iteration 4, inertia 8824.832909813438
Iteration 5, inertia 8819.837479151136
Iteration 6, inertia 8819.10078183813
Iteration 7, inertia 8819.035876469317
Iteration 8, inertia 8818.817563005117
Iteration 9, inertia 8818.646974003661
Iteration 10, inertia 8818.289586515795
Iteration 11, inertia 8818.159940170133
Iteration 12, inertia 8817.98907376872
Iteration 13, inertia 8817.865936421687
Iteration 14, inertia 8817.806917535887
Iteration 15, inertia 8817.776092265041
Iteration 16, inertia 8817.693711678381
Iteration 17, inertia 8817.668602183467
Converged at iteration 17: strict convergence.
Initialization complete
Iteration 0, inertia 14045.247730533238
Iteration 1, inertia 9577.932151433946
Iteration 2, inertia 9503.316242864512
Iteration 3, inertia 9481.915359950877
Iteration 4, inertia 9463.72742189162
Iteration 5, inertia 9441.526557119001
Iteration 6, inertia 9420.35228646382
Iteration 7, inertia 9406.373961589723
Iteration 8, inertia 9402.970170630471
Iteration 9, inertia 9400.047385253652
Iteration 10, inertia 9399.555304602607
Iteration 11, inertia 9399.498839497075
Iteration 12, inertia 9399.41018122203
Iteration 13, inertia 9399.298956532946
Iteration 14, inertia 9399.274680230332
Iteration 15, inertia 9399.252245628535
Converged at iteration 15: strict convergence.
Initialization complete
Iteration 0, inertia 13317.99475078725
Iteration 1, inertia 8977.851327856566
Iteration 2, inertia 8895.604119959466
Iteration 3, inertia 8873.687473142223
Iteration 4, inertia 8849.570415401187
Iteration 5, inertia 8805.858315057407
Iteration 6, inertia 8780.421553985545
Iteration 7, inertia 8775.53080855752
Iteration 8, inertia 8772.61887038512
Iteration 9, inertia 8769.175262861294
Iteration 10, inertia 8768.382144887562
Iteration 11, inertia 8767.704981473666
Iteration 12, inertia 8766.274753740365
Iteration 13, inertia 8765.417169729853
Iteration 14, inertia 8765.398780834916
Converged at iteration 14: strict convergence.
```

Out[27]:
```
KMeans(max_iter=500, n_clusters=3, verbose=1)
```

In [28]:
```
clusters3 = kmeans3.predict(df_min_max_scaled)
```

In [29]:
```
size3 = cluster_sizes(clusters3)

for c in size3.keys():
    print("Size of Cluster", c, "= ", size3[c])
```

```
Size of Cluster 0 =  1058
Size of Cluster 1 =  636
Size of Cluster 2 =  417
```

In [30]:
```
# View centroids for an aggregate representation and a characterization of each
pd.options.display.float_format='{:,.2f}'.format

centroids3 = pd.DataFrame(kmeans3.cluster_centers_, columns=df_min_max_scaled.co
centroids3
```

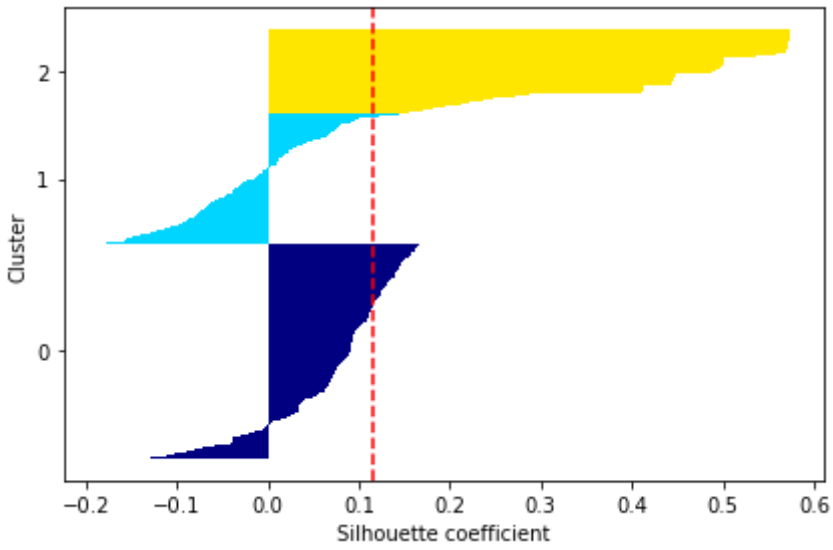
Out[30]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	family
0	0.22	0.58	0.39	-0.00	1.00	0.14	
1	0.21	0.33	0.19	0.99	0.01	0.34	
2	0.19	0.43	0.52	0.99	0.01	0.04	

In [31]:
```
# Silhouette Analysis at n = 3:
c3_silhouette = metrics.silhouette_samples(df_min_max_scaled, clusters3)
print('Mean Silhouette Value :', c3_silhouette.mean())
```

```
Mean Silhouette Value : 0.11634874352766442
```

In [32]:
```
# Plot and Evaluate the Silhouettes:
plot_silhouettes(df_min_max_scaled, clusters3)
```



Above, shows the results of the silhouette analysis for K=3, which reveals that the algorithm performed neither better nor worse than at K = 5. The plot of the silhouettes shows that cluster

2 outperformed the other clusters with all its coefficients above the mean silhouette value. Cluster 1 performed the worst and did not have any coefficients above the mean silhouette value, but instead has negative coefficients. When evaluating the centroids, cluster 0 has Gender_Male with a value of 1.00 and Gender_Female with a value of 0. Cluster 0 most likely represents the male gender. Cluster 1 and 2 both contain a value of 0.99 for Gender_Female and 0.01 for Gender_Male, which shows that most likely Cluster 1 is misclassified. Most likely this cluster is pulling coefficients where it should not be and is too close to cluster 0 to be its own cluster. We can conclude from the silhouette plots that likely three cluster is still too high and that two clusters may be sufficient.

In [33]:
```
kmeans2 = KMeans(n_clusters=2, max_iter=500, verbose=1) # k-means with n = 2
```

In [34]:
```
kmeans2.fit(df_min_max_scaled)
```

```
Initialization complete
Iteration 0, inertia 21141.470043487447
Iteration 1, inertia 10146.649638459037
Iteration 2, inertia 10052.393229584075
Iteration 3, inertia 9978.55949142651
Iteration 4, inertia 9931.576115637514
Iteration 5, inertia 9882.008890876072
Iteration 6, inertia 9833.033569152649
Iteration 7, inertia 9820.76522415991
Iteration 8, inertia 9810.381012107375
Iteration 9, inertia 9806.43745895125
Iteration 10, inertia 9802.52445906197
Iteration 11, inertia 9788.188974953495
Iteration 12, inertia 9784.590860736775
Iteration 13, inertia 9781.587461096071
Iteration 14, inertia 9764.62868015089
Iteration 15, inertia 9703.250240710777
Iteration 16, inertia 9654.064419604409
Iteration 17, inertia 9640.924590400316
Iteration 18, inertia 9633.269288654592
Iteration 19, inertia 9627.22637779535
Iteration 20, inertia 9626.211067157712
Iteration 21, inertia 9625.966275262466
Iteration 22, inertia 9625.900707169734
Iteration 23, inertia 9625.880356300455
Converged at iteration 23: strict convergence.
Initialization complete
Iteration 0, inertia 16459.557174738147
Iteration 1, inertia 9900.362639271163
Iteration 2, inertia 9489.377342057842
Iteration 3, inertia 9440.231621454297
Iteration 4, inertia 9439.746651906471
Iteration 5, inertia 9439.70314554615
Converged at iteration 5: strict convergence.
Initialization complete
Iteration 0, inertia 16654.425161503892
Iteration 1, inertia 9928.457371508537
Iteration 2, inertia 9838.23752341025
Iteration 3, inertia 9788.863247705292
Iteration 4, inertia 9548.861615101538
Iteration 5, inertia 9442.673615150014
Iteration 6, inertia 9439.835639639505
Iteration 7, inertia 9439.703145546147
Converged at iteration 7: strict convergence.
```

```
Initialization complete
Iteration 0, inertia 18427.311216298647
Iteration 1, inertia 10184.320163749075
Iteration 2, inertia 10006.009344174558
Iteration 3, inertia 9946.739537737863
Iteration 4, inertia 9924.402102243388
Iteration 5, inertia 9910.780896241331
Iteration 6, inertia 9905.110900156222
Iteration 7, inertia 9902.264415823509
Iteration 8, inertia 9897.72566174957
Iteration 9, inertia 9895.57672400781
Iteration 10, inertia 9894.774416642193
Iteration 11, inertia 9892.907792732634
Iteration 12, inertia 9889.535477719926
Iteration 13, inertia 9885.189090329697
Iteration 14, inertia 9882.336535503198
Iteration 15, inertia 9878.48056558997
Iteration 16, inertia 9874.201130739495
Iteration 17, inertia 9869.024206385815
Iteration 18, inertia 9859.9664768237
Iteration 19, inertia 9851.6930077618
Iteration 20, inertia 9844.315605926562
Iteration 21, inertia 9838.560305605188
Iteration 22, inertia 9828.42281943376
Iteration 23, inertia 9813.08654486762
Iteration 24, inertia 9801.820301212163
Iteration 25, inertia 9796.444293037726
Iteration 26, inertia 9794.424054751174
Iteration 27, inertia 9794.231982639427
Iteration 28, inertia 9794.184865647467
Converged at iteration 28: center shift 1.1769791973055172e-05 within tolerance
1.1825910645989139e-05.
Initialization complete
Iteration 0, inertia 16402.36648533896
Iteration 1, inertia 9994.565703554776
Iteration 2, inertia 9934.77322692749
Iteration 3, inertia 9909.912786027671
Iteration 4, inertia 9883.607078127476
Iteration 5, inertia 9850.433915715637
Iteration 6, inertia 9791.58073461284
Iteration 7, inertia 9734.449876536686
Iteration 8, inertia 9646.394815196176
Iteration 9, inertia 9627.336127098832
Iteration 10, inertia 9625.954342511643
Iteration 11, inertia 9625.884133923937
Iteration 12, inertia 9625.86108434419
Iteration 13, inertia 9625.840681351561
Converged at iteration 13: strict convergence.
Initialization complete
Iteration 0, inertia 14969.584620698479
Iteration 1, inertia 9901.195321252426
Iteration 2, inertia 9853.041301002952
Iteration 3, inertia 9828.346078346132
Iteration 4, inertia 9821.497722659567
Iteration 5, inertia 9816.797972478129
Iteration 6, inertia 9815.160587833674
Iteration 7, inertia 9813.203694536212
Iteration 8, inertia 9812.379447087993
Iteration 9, inertia 9811.952753836797
Iteration 10, inertia 9811.747610338476
Iteration 11, inertia 9811.640620134325
Iteration 12, inertia 9811.132844140293
Iteration 13, inertia 9810.651492658839
Iteration 14, inertia 9810.461157499503
Iteration 15, inertia 9809.569339539601
```

```
Iteration 16, inertia 9808.659582332328
Iteration 17, inertia 9808.2561123607
Iteration 18, inertia 9807.986336248636
Iteration 19, inertia 9807.906534765058
Iteration 20, inertia 9807.487360220597
Iteration 21, inertia 9806.667738269354
Iteration 22, inertia 9805.834740565373
Iteration 23, inertia 9800.048920420628
Iteration 24, inertia 9788.128577786762
Iteration 25, inertia 9779.417579204792
Iteration 26, inertia 9775.38432414417
Iteration 27, inertia 9761.684802413005
Iteration 28, inertia 9709.236797762258
Iteration 29, inertia 9653.799784178347
Iteration 30, inertia 9642.172114603334
Iteration 31, inertia 9636.202113008087
Iteration 32, inertia 9620.09415505929
Iteration 33, inertia 9617.977388582693
Iteration 34, inertia 9615.658558431482
Iteration 35, inertia 9610.65542615645
Iteration 36, inertia 9607.248540288658
Iteration 37, inertia 9603.493341174244
Iteration 38, inertia 9599.523654632054
Iteration 39, inertia 9597.642822643924
Iteration 40, inertia 9596.209483776676
Iteration 41, inertia 9595.35635092364
Iteration 42, inertia 9594.882428291552
Iteration 43, inertia 9594.678998923426
Iteration 44, inertia 9594.37636625157
Iteration 45, inertia 9594.115982262558
Iteration 46, inertia 9594.001524563828
Iteration 47, inertia 9592.870319464864
Iteration 48, inertia 9587.715138968875
Iteration 49, inertia 9578.793233518765
Iteration 50, inertia 9556.18315100723
Iteration 51, inertia 9517.365630351003
Iteration 52, inertia 9457.17665798308
Iteration 53, inertia 9440.017782540526
Iteration 54, inertia 9439.746651906471
Iteration 55, inertia 9439.703145546147
Converged at iteration 55: strict convergence.
Initialization complete
Iteration 0, inertia 16595.25826599754
Iteration 1, inertia 10109.37538869094
Iteration 2, inertia 9915.261803411886
Iteration 3, inertia 9789.099147804835
Iteration 4, inertia 9778.126542918279
Iteration 5, inertia 9777.915684703536
Converged at iteration 5: strict convergence.
Initialization complete
Iteration 0, inertia 15375.183674102089
Iteration 1, inertia 9952.678493621155
Iteration 2, inertia 9884.256882726944
Iteration 3, inertia 9864.875145644339
Iteration 4, inertia 9859.869224463644
Iteration 5, inertia 9858.379095829157
Iteration 6, inertia 9854.219459620635
Iteration 7, inertia 9841.672028071058
Iteration 8, inertia 9832.676859254072
Iteration 9, inertia 9818.161920484818
Iteration 10, inertia 9805.876930142955
Iteration 11, inertia 9798.29221813926
Iteration 12, inertia 9794.467009896513
Iteration 13, inertia 9794.22218892144
Iteration 14, inertia 9794.184865647467
```

```
Converged at iteration 14: center shift 1.1769791973055237e-05 within tolerance
1.1825910645989139e-05.
Initialization complete
Iteration 0, inertia 18359.985955431566
Iteration 1, inertia 10101.593413896659
Iteration 2, inertia 9982.754190681128
Iteration 3, inertia 9917.840303560613
Iteration 4, inertia 9890.496715960406
Iteration 5, inertia 9876.352494100358
Iteration 6, inertia 9865.682282638423
Iteration 7, inertia 9853.175178022406
Iteration 8, inertia 9841.405257717575
Iteration 9, inertia 9828.743052973643
Iteration 10, inertia 9812.10020465191
Iteration 11, inertia 9800.966469728079
Iteration 12, inertia 9795.474672343822
Iteration 13, inertia 9794.39208071576
Iteration 14, inertia 9794.247097906948
Iteration 15, inertia 9794.203342487108
Converged at iteration 15: center shift 1.1189193902305637e-05 within tolerance
1.1825910645989139e-05.
Initialization complete
Iteration 0, inertia 14840.19634222251
Iteration 1, inertia 10012.892987021458
Iteration 2, inertia 9937.527748973907
Iteration 3, inertia 9874.816695079277
Iteration 4, inertia 9840.194716250331
Iteration 5, inertia 9814.311264293689
Iteration 6, inertia 9800.700957677194
Iteration 7, inertia 9771.602530465227
Iteration 8, inertia 9541.236198237983
Iteration 9, inertia 9441.157699191357
Iteration 10, inertia 9439.835639639505
Iteration 11, inertia 9439.70314554615
Converged at iteration 11: strict convergence.
```

Out[34]:
```
KMeans(max_iter=500, n_clusters=2, verbose=1)
```

In [35]:
```
clusters2 = kmeans2.predict(df_min_max_scaled)
```

In [36]:
```
size2 = cluster_sizes(clusters2)

for c in size2.keys():
    print("Size of Cluster", c, "= ", size2[c])
```

```
Size of Cluster 0 =  1067
Size of Cluster 1 =  1044
```

In [37]:
```
# View centroids for an aggregate representation and a characterization of each
pd.options.display.float_format='{:,.2f}'.format

centroids2 = pd.DataFrame(kmeans2.cluster_centers_, columns=df_min_max_scaled.co
centroids2
```

Out[37]:

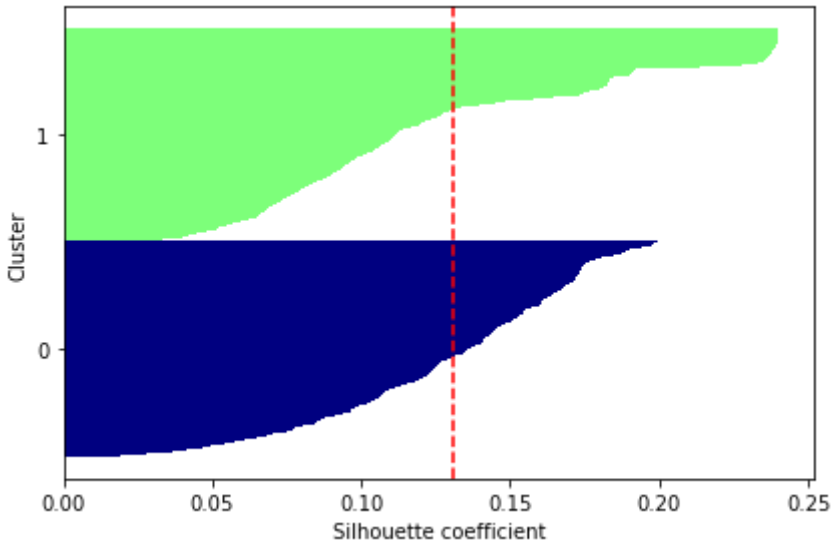
	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	family
0	0.22	0.58	0.39	-0.00	1.00	0.14	

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	family
1	0.21	0.36	0.32	1.00	0.00	0.22	

In [38]:
```
# Silhouette Analysis at n = 2:
c2_silhouette = metrics.silhouette_samples(df_min_max_scaled, clusters2)
print('Mean Silhouette Value :', c2_silhouette.mean())
```

```
Mean Silhouette Value : 0.13093478332005926
```

In [39]:
```
# Plot and Evaluate the Silhouettes:
plot_silhouettes(df_min_max_scaled, clusters2)
```



above shows the results of the silhouette analysis for K=2, which achieved the best silhouette plot compared to previous plots at K = 5 and K = 3. This silhouette plot shows that both cluster 0 and 1 have coefficients that are above the mean silhouette value and none of the coefficients are negative. Both clusters are neither thick nor full, although, cluster 0 appears thicker than cluster 1, but from the clustering results above, this result is most successful. When looking at the centroids, the two features that stand out that most likely represent the clusters compared to all other features is Gender_Male and Gender_Female. In cluster 0, Gender_Male has a value of 1.00 while Gender_Female has a value of -0.00 and in cluster 1, Gender_Female has a value of 1.00 while Gender_Male has a value 0.00. Moreover, we can conclude from the silhouette plots above that likely, cluster 0 represents males and cluster 1 represents female. This evaluation shows that a pattern exists by gender and that gender may play a role in the dataset and in determining classification of obesity levels.

Next, we will create age groups and seperate the age of each individual based on generation. Exploring age groups will allow us to re-evaluate the clusters and determine if a pattern exists also within age group for classification.

Discretize the Age attribute into 4 seperate age groups and re-run K-Means Clustering:

Gen-Z (1997 – 2012), Age: 9 – 24

Millennials (1981 – 1996), Age: 25 – 40

Gen-X (1965 – 1980), Age: 41 – 56

Boomers (1955 - 1964), Age: 57 - 66

In [40]:
```
data_numeric.Age.min() #youngest age in the dataset
```

Out[40]:
```
14
```

In [41]:
```
data_numeric.Age.max() #oldest age in the dataset
```

Out[41]:
```
61
```

In [42]:
```
age_bins = pd.qcut(data_numeric.Age, [0, .61, .972, 1])
age_bins.head(5)
```

Out[42]:
```
0    (13.999, 24.0]
1    (13.999, 24.0]
2    (13.999, 24.0]
3      (24.0, 40.0]
4    (13.999, 24.0]
Name: Age, dtype: category
Categories (3, interval[float64]): [(13.999, 24.0] < (24.0, 40.0] < (40.0, 61.
0]]
```

In [43]:
```
age_bins = pd.qcut(data_numeric.Age, [0, .61, .972, 1], labels = ['Gen-Z', 'Mill
age_df = pd.concat([age_bins, df2['Age']], axis=1)
age_df.columns = ['Age Group', 'Age']
age_df.head(10)
```

Out[43]:

	Age Group	Age
0	Gen-Z	21
1	Gen-Z	21
2	Gen-Z	23
3	Millenials	27
4	Gen-Z	22
5	Millenials	29
6	Gen-Z	23
7	Gen-Z	22
8	Gen-Z	24
9	Gen-Z	22

In [44]:
```
data_age_groups = data_numeric
data_age_groups["Age"] = age_df['Age Group']
```

In [45]:
```
data_age_groups.head(10)
```

Out[45]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	fa
0	Gen-Z	1.62	64.00	1	0	0	
1	Gen-Z	1.52	56.00	1	0	0	
2	Gen-Z	1.80	77.00	0	1	0	
3	Millenials	1.80	87.00	0	1	1	
4	Gen-Z	1.78	89.80	0	1	1	
5	Millenials	1.62	53.00	0	1	1	
6	Gen-Z	1.50	55.00	1	0	0	
7	Gen-Z	1.64	53.00	0	1	1	
8	Gen-Z	1.78	64.00	0	1	0	
9	Gen-Z	1.72	68.00	0	1	0	

In [46]:
```
# Create Dummy Variables for Binned Dataset:
df_age_groups = pd.get_dummies(data_age_groups)
df_age_groups.head(5)
```

Out[46]:

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	family_histo
0	1.62	64.00	1	0	0	
1	1.52	56.00	1	0	0	
2	1.80	77.00	0	1	0	
3	1.80	87.00	0	1	1	
4	1.78	89.80	0	1	1	

K-means algorithm with the three generational age groups: Gen-Z, Millennials, and Gen-X and Boomers. This exploration is being explored to see if a pattern exists based on age range which the cluster analysis for the full dataset did not evaluate since the age groups were not grouped into categories. The youngest age is 14 and the oldest age is 61. The age groups are created by binning the Age attribute and then transforming the age group attribute into dummy variables. For exploratory purposes, K-means is performed on the dataset first without min-max normalization and second with min-max normalization at K = 3.

In [47]:
```
# Perform K-Means Clustering with N = 3:
kmeans = KMeans(n_clusters=3, max_iter=500, verbose=1) #initialize k-means with
```

In [48]:
```
kmeans.fit(df_age_groups)
```

```
Initialization complete
Iteration 0, inertia 266216.09575717594
Iteration 1, inertia 215521.27371160412
Iteration 2, inertia 210755.68046562248
Iteration 3, inertia 209722.10499704749
Iteration 4, inertia 209418.75170585237
Iteration 5, inertia 209024.46758980726
Iteration 6, inertia 208957.87320545508
Iteration 7, inertia 208952.94787903182
Converged at iteration 7: center shift 0.0008306717121226465 within tolerance 0.
0015358503717230955.
Initialization complete
Iteration 0, inertia 502841.1735953951
Iteration 1, inertia 327560.09540897014
Iteration 2, inertia 315527.1447145528
Iteration 3, inertia 309720.53549766104
Iteration 4, inertia 300890.08942123153
Iteration 5, inertia 292303.3695465401
Iteration 6, inertia 287217.5099774456
Iteration 7, inertia 283800.36697429675
Iteration 8, inertia 278425.3310745159
Iteration 9, inertia 270545.8711724134
Iteration 10, inertia 246805.06029650217
Iteration 11, inertia 232750.70198980303
Iteration 12, inertia 226653.87489398956
Iteration 13, inertia 216920.98018306002
Iteration 14, inertia 212292.1259899532
Iteration 15, inertia 209925.16585043436
Iteration 16, inertia 209641.8566251863
Iteration 17, inertia 209612.29209838895
Iteration 18, inertia 209601.88285883892
Iteration 19, inertia 209598.80247956378
Converged at iteration 19: strict convergence.
Initialization complete
Iteration 0, inertia 326936.9440904384
Iteration 1, inertia 226724.44726938492
Iteration 2, inertia 213994.4657317486
Iteration 3, inertia 210480.97315506768
Iteration 4, inertia 209660.8099794086
Iteration 5, inertia 209614.6304476699
Iteration 6, inertia 209601.88285883892
Iteration 7, inertia 209598.80247956383
Converged at iteration 7: strict convergence.
Initialization complete
Iteration 0, inertia 244660.38351519656
Iteration 1, inertia 211950.75265903442
Iteration 2, inertia 209849.7543072008
Iteration 3, inertia 209518.7500452044
Iteration 4, inertia 209060.11326694212
Iteration 5, inertia 208957.87320545508
Iteration 6, inertia 208952.94787903182
Converged at iteration 6: center shift 0.0008306717121226465 within tolerance 0.
0015358503717230955.
Initialization complete
Iteration 0, inertia 293038.3352946884
Iteration 1, inertia 213135.03302011758
Iteration 2, inertia 210104.25819821077
Iteration 3, inertia 209618.1187074107
Iteration 4, inertia 209113.16767690537
Iteration 5, inertia 208962.37801962328
Iteration 6, inertia 208952.94787903185
```

```
Converged at iteration 6: center shift 0.0008306717121226769 within tolerance 0.
0015358503717230955.
Initialization complete
Iteration 0, inertia 275553.3335700919
Iteration 1, inertia 222295.79489500792
Iteration 2, inertia 212492.7176855583
Iteration 3, inertia 209869.48135591563
Iteration 4, inertia 209628.24667814613
Iteration 5, inertia 209612.29209838895
Iteration 6, inertia 209601.88285883892
Iteration 7, inertia 209598.80247956383
Converged at iteration 7: strict convergence.
Initialization complete
Iteration 0, inertia 286761.1414317467
Iteration 1, inertia 210695.01280865865
Iteration 2, inertia 209668.3036758006
Iteration 3, inertia 209614.6304476699
Iteration 4, inertia 209601.88285883892
Iteration 5, inertia 209598.80247956383
Converged at iteration 5: strict convergence.
Initialization complete
Iteration 0, inertia 383050.70537416794
Iteration 1, inertia 268032.44644019724
Iteration 2, inertia 222407.19095729562
Iteration 3, inertia 213626.56770579072
Iteration 4, inertia 210456.74923156487
Iteration 5, inertia 209703.02658441686
Iteration 6, inertia 209362.03202101542
Iteration 7, inertia 209013.56709803338
Iteration 8, inertia 208954.93374886544
Iteration 9, inertia 208952.94787903185
Converged at iteration 9: center shift 0.0008306717121226467 within tolerance 0.
0015358503717230955.
Initialization complete
Iteration 0, inertia 374254.3643704856
Iteration 1, inertia 227441.0113142989
Iteration 2, inertia 210113.7182635559
Iteration 3, inertia 209144.6971247129
Iteration 4, inertia 209037.23080006722
Iteration 5, inertia 208970.75843448716
Iteration 6, inertia 208958.73649992305
Converged at iteration 6: center shift 0.0010520324381042456 within tolerance 0.
0015358503717230955.
Initialization complete
Iteration 0, inertia 306059.6127144407
Iteration 1, inertia 230381.17640096927
Iteration 2, inertia 216298.96923962721
Iteration 3, inertia 211115.72555699918
Iteration 4, inertia 209886.61690318544
Iteration 5, inertia 209525.64261090878
Iteration 6, inertia 209040.15129423398
Iteration 7, inertia 208957.87320545508
Iteration 8, inertia 208952.94787903182
Converged at iteration 8: center shift 0.0008306717121226166 within tolerance 0.
0015358503717230955.
```

Out[48]:
```
KMeans(max_iter=500, n_clusters=3, verbose=1)
```

In [49]:
```
age_clusters = kmeans.predict(df_age_groups)
```

In [50]:
```
size = cluster_sizes(age_clusters)
```

```
for c in size.keys():
    print("Size of Cluster", c, "= ", size[c])
```

```
Size of Cluster 0 =  789
Size of Cluster 1 =  731
Size of Cluster 2 =  591
```

In [51]:
```
# View centroids for an aggregate representation and a characterization of each
pd.options.display.float_format='{:,.2f}'.format

centroids = pd.DataFrame(kmeans.cluster_centers_, columns=df_age_groups.columns.
centroids
```

Out[51]:

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	family_histo
0	1.70	82.18	0.39	0.61	0.13	
1	1.74	116.59	0.44	0.56	0.00	
2	1.65	55.37	0.69	0.31	0.48	

In [52]:
```
centroids['Age_Gen-Z'] #clusters containing Gen-Z
```

Out[52]:
```
0   0.64
1   0.43
2   0.90
Name: Age_Gen-Z, dtype: float64
```

In [53]:
```
centroids['Age_Millenials'] #clusters containing Millenials
```

Out[53]:
```
0   0.31
1   0.56
2   0.10
Name: Age_Millenials, dtype: float64
```

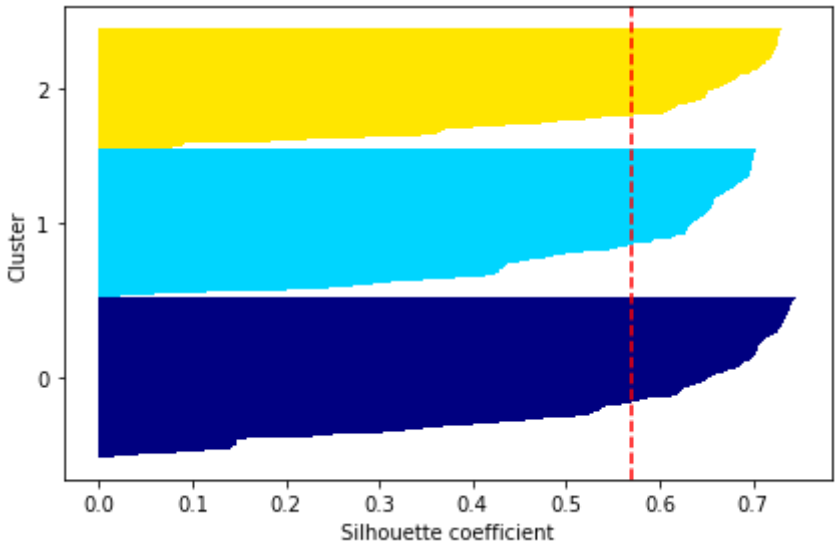
In [54]:
```
centroids['Age_Gen-X & Boomers'] #clusters containing Gen-X and Boomers
```

Out[54]:
```
0   0.04
1   0.01
2   0.00
Name: Age_Gen-X & Boomers, dtype: float64
```

In [55]:
```
# Silhouette Analysis at n = 3:
age_silhouette = metrics.silhouette_samples(df_age_groups, age_clusters)
print('Mean Silhouette Value :', age_silhouette.mean())
```

```
Mean Silhouette Value : 0.5691256560102319
```

In [56]:
```
# Plot and Evaluate the Silhouettes:
plot_silhouettes(df_age_groups, age_clusters)
```



The results of cluster analysis without normalization shows a very healthy silhouette plot with all three clusters full, thick, and with coefficients above the mean silhouette value. Figure 2.1 below confirms that clusters when age is grouped by range. When looking at the centroids, cluster 2 shows Gen-Z at 0.9 while Millennials at .10 and Gen-X and Boomers at 0.00. Most likely Gen-Z is represented in cluster 2.

In [57]:
```
# Calculate Completeness and Homogeneity for the clusters:
complete = completeness_score(labels_num, age_clusters)
print(f"Completeness Score for Clusters: {complete}")
homogene = homogeneity_score(labels_num, age_clusters)
print(f"Homogeneity Score for Clusters: {homogene}")
```

```
Completeness Score for Clusters: 0.7020884578966542
Homogeneity Score for Clusters: 0.39448267211636195
```

The completeness and homogeneity scores were calculated for clusters since the class labels exist for further examination of the cluster quality. The completeness score was 0.70 which shows that members of a given class are assigned to the same cluster 70% of the time. The completeness score is positive and confirms that the clusters captured most of one class. The homogeneity score was much lower at 0.39 which shows that the clusters are not pure. These results may indicate that age group may be a factor in deciding the clusters for the data, but it may not be the main factor that affects obesity level for classification. The silhouette plots above display that a pattern exist but we must take into consideration that the data was not scaled. As such, we will next, perform K-means again with the data normalized to validate the results.

Perform K-Means with Normalized Data on Age Groups for Comparsion:

In [58]:
```
# Normalize the dataset with Min-Max Scaling:
df_age_groups_norm = df_age_groups.copy()
for column in df_age_groups_norm.columns:
    df_age_groups_norm[column] = (df_age_groups_norm[column] - df_age_groups_nor
```

In [59]:
```
# View normalized data:
```

```
print(df_age_groups_norm)
```

```
      Height  Weight  Gender_Female  Gender_Male  \
0       0.32    0.19           1.00         0.00
1       0.13    0.13           1.00         0.00
2       0.66    0.28           0.00         1.00
3       0.66    0.36           0.00         1.00
4       0.62    0.38           0.00         1.00
...      ...     ...            ...          ...
2106    0.49    0.69           1.00         0.00
2107    0.56    0.71           1.00         0.00
2108    0.57    0.71           1.00         0.00
2109    0.55    0.70           1.00         0.00
2110    0.54    0.71           1.00         0.00

      family_history_with_overweight_no  family_history_with_overweight_yes  \
0                                  0.00                                1.00
1                                  0.00                                1.00
2                                  0.00                                1.00
3                                  1.00                                0.00
4                                  1.00                                0.00
...                                 ...                                 ...
2106                               0.00                                1.00
2107                               0.00                                1.00
2108                               0.00                                1.00
2109                               0.00                                1.00
2110                               0.00                                1.00

      FAVC_no  FAVC_yes  FCVC_Always  FCVC_Never  FCVC_Sometimes  NCP_1  \
0        1.00      0.00         0.00        0.00            1.00   0.00
1        1.00      0.00         1.00        0.00            0.00   0.00
2        1.00      0.00         0.00        0.00            1.00   0.00
3        1.00      0.00         1.00        0.00            0.00   0.00
4        1.00      0.00         0.00        0.00            1.00   1.00
...       ...       ...          ...         ...             ...    ...
2106     0.00      1.00         1.00        0.00            0.00   0.00
2107     0.00      1.00         1.00        0.00            0.00   0.00
2108     0.00      1.00         1.00        0.00            0.00   0.00
2109     0.00      1.00         1.00        0.00            0.00   0.00
2110     0.00      1.00         1.00        0.00            0.00   0.00

      NCP_2  NCP_3  NCP_3+  CAEC_Always  CAEC_Frequently  CAEC_Sometimes  \
0      0.00   1.00    0.00         0.00             0.00            1.00
1      0.00   1.00    0.00         0.00             0.00            1.00
2      0.00   1.00    0.00         0.00             0.00            1.00
3      0.00   1.00    0.00         0.00             0.00            1.00
4      0.00   0.00    0.00         0.00             0.00            1.00
...     ...    ...     ...          ...              ...             ...
2106   0.00   1.00    0.00         0.00             0.00            1.00
2107   0.00   1.00    0.00         0.00             0.00            1.00
2108   0.00   1.00    0.00         0.00             0.00            1.00
2109   0.00   1.00    0.00         0.00             0.00            1.00
2110   0.00   1.00    0.00         0.00             0.00            1.00

      CAEC_no  SMOKE_no  SMOKE_yes  CH2O_Between 1 and 2 L  \
0        0.00      1.00       0.00                    1.00
1        0.00      0.00       1.00                    0.00
2        0.00      1.00       0.00                    1.00
3        0.00      1.00       0.00                    1.00
4        0.00      1.00       0.00                    1.00
...       ...       ...        ...                     ...
2106     0.00      1.00       0.00                    0.00
2107     0.00      1.00       0.00                    1.00
2108     0.00      1.00       0.00                    1.00
2109     0.00      1.00       0.00                    1.00
```

```
2110     0.00      1.00       0.00                    1.00

      CH2O_Less than a liter  CH2O_More than 2 L  SCC_no  SCC_yes  \
0                       0.00                0.00    1.00     0.00
1                       0.00                1.00    0.00     1.00
2                       0.00                0.00    1.00     0.00
3                       0.00                0.00    1.00     0.00
4                       0.00                0.00    1.00     0.00
...                      ...                 ...     ...      ...
2106                    1.00                0.00    1.00     0.00
2107                    0.00                0.00    1.00     0.00
2108                    0.00                0.00    1.00     0.00
2109                    0.00                0.00    1.00     0.00
2110                    0.00                0.00    1.00     0.00

      FAF_1 or 2 days  FAF_2 or 4 days  FAF_4 or 5 days  FAF_I do not have  \
0                0.00             0.00             0.00               1.00
1                0.00             0.00             1.00               0.00
2                0.00             1.00             0.00               0.00
3                0.00             1.00             0.00               0.00
4                0.00             0.00             0.00               1.00
...               ...              ...              ...                ...
2106             1.00             0.00             0.00               0.00
2107             1.00             0.00             0.00               0.00
2108             1.00             0.00             0.00               0.00
2109             1.00             0.00             0.00               0.00
2110             1.00             0.00             0.00               0.00

      TUE_0-2 Hours  TUE_3-5 Hours  TUE_More than 5 Hours  CALC_Always  \
0              0.00           1.00                   0.00         0.00
1              1.00           0.00                   0.00         0.00
2              0.00           1.00                   0.00         0.00
3              1.00           0.00                   0.00         0.00
4              1.00           0.00                   0.00         0.00
...             ...            ...                    ...          ...
2106           1.00           0.00                   0.00         0.00
2107           1.00           0.00                   0.00         0.00
2108           1.00           0.00                   0.00         0.00
2109           1.00           0.00                   0.00         0.00
2110           1.00           0.00                   0.00         0.00

      CALC_Frequently  CALC_Sometimes  CALC_no  MTRANS_Automobile  \
0                0.00            0.00     1.00               0.00
1                0.00            1.00     0.00               0.00
2                1.00            0.00     0.00               0.00
3                1.00            0.00     0.00               0.00
4                0.00            1.00     0.00               0.00
...               ...             ...      ...                ...
2106             0.00            1.00     0.00               0.00
2107             0.00            1.00     0.00               0.00
2108             0.00            1.00     0.00               0.00
2109             0.00            1.00     0.00               0.00
2110             0.00            1.00     0.00               0.00

      MTRANS_Bike  MTRANS_Motorbike  MTRANS_Public_Transportation  \
0            0.00              0.00                          1.00
1            0.00              0.00                          1.00
2            0.00              0.00                          1.00
3            0.00              0.00                          0.00
4            0.00              0.00                          1.00
...           ...               ...                           ...
2106         0.00              0.00                          1.00
2107         0.00              0.00                          1.00
2108         0.00              0.00                          1.00
2109         0.00              0.00                          1.00
```

```
2110         0.00              0.00                          1.00

      MTRANS_Walking  Age_Gen-Z  Age_Millenials  Age_Gen-X & Boomers
0               0.00       1.00            0.00                 0.00
1               0.00       1.00            0.00                 0.00
2               0.00       1.00            0.00                 0.00
3               1.00       0.00            1.00                 0.00
4               0.00       1.00            0.00                 0.00
...              ...        ...             ...                  ...
2106            0.00       1.00            0.00                 0.00
2107            0.00       1.00            0.00                 0.00
2108            0.00       1.00            0.00                 0.00
2109            0.00       1.00            0.00                 0.00
2110            0.00       1.00            0.00                 0.00

[2111 rows x 45 columns]
```

In [60]:
```
# Perform K-Means Clustering with N = 3:
kmeans3 = KMeans(n_clusters=3, max_iter=500, verbose=1)
```

In [61]:
```
kmeans3.fit(df_age_groups_norm)
```

```
Initialization complete
Iteration 0, inertia 15293.30577388341
Iteration 1, inertia 9968.066331837832
Iteration 2, inertia 9920.180131785271
Iteration 3, inertia 9903.77236856074
Iteration 4, inertia 9896.176445738252
Iteration 5, inertia 9891.495588792406
Iteration 6, inertia 9889.624965572522
Iteration 7, inertia 9888.823257389276
Iteration 8, inertia 9888.487377405678
Iteration 9, inertia 9888.340285312179
Iteration 10, inertia 9888.20252958357
Iteration 11, inertia 9887.532993038576
Iteration 12, inertia 9887.504196402364
Converged at iteration 12: strict convergence.
Initialization complete
Iteration 0, inertia 15755.748152264905
Iteration 1, inertia 10291.16329342617
Iteration 2, inertia 10228.905860318164
Iteration 3, inertia 10188.107451127948
Iteration 4, inertia 10144.37857128053
Iteration 5, inertia 10107.001253617971
Iteration 6, inertia 10086.05688167689
Iteration 7, inertia 10077.969028399091
Iteration 8, inertia 10068.115905357135
Iteration 9, inertia 10054.640327576219
Iteration 10, inertia 10036.68992669457
Iteration 11, inertia 10012.308155223136
Iteration 12, inertia 9995.089501496252
Iteration 13, inertia 9979.257157992513
Iteration 14, inertia 9938.759373638944
Iteration 15, inertia 9911.724132573296
Iteration 16, inertia 9891.1788188847
Iteration 17, inertia 9754.311626803314
Iteration 18, inertia 9678.154213662237
Iteration 19, inertia 9667.068511677915
Iteration 20, inertia 9660.214612839805
Iteration 21, inertia 9654.574790207527
Iteration 22, inertia 9645.63794782316
Iteration 23, inertia 9644.899891129677
```

```
Iteration 24, inertia 9644.748605586845
Iteration 25, inertia 9644.595919167177
Iteration 26, inertia 9644.549083693555
Iteration 27, inertia 9641.042116346523
Iteration 28, inertia 9638.110099852393
Iteration 29, inertia 9634.979187430297
Iteration 30, inertia 9632.762887140108
Iteration 31, inertia 9632.086194242833
Iteration 32, inertia 9631.65577226331
Iteration 33, inertia 9631.569838198466
Iteration 34, inertia 9631.539746213291
Converged at iteration 34: strict convergence.
Initialization complete
Iteration 0, inertia 14748.903594063975
Iteration 1, inertia 9965.186003070134
Iteration 2, inertia 9825.169923880945
Iteration 3, inertia 9744.272576918649
Iteration 4, inertia 9667.587922972407
Iteration 5, inertia 9644.10573691932
Iteration 6, inertia 9635.267693732794
Iteration 7, inertia 9633.29914147962
Iteration 8, inertia 9632.952141350772
Iteration 9, inertia 9632.87014898023
Converged at iteration 9: strict convergence.
Initialization complete
Iteration 0, inertia 14484.898807356742
Iteration 1, inertia 9955.81303074818
Iteration 2, inertia 9713.282859132194
Iteration 3, inertia 9671.355873128363
Iteration 4, inertia 9650.513068725491
Iteration 5, inertia 9645.317685969698
Iteration 6, inertia 9642.530506536332
Iteration 7, inertia 9641.485885635826
Iteration 8, inertia 9640.28240915911
Iteration 9, inertia 9639.857937703955
Iteration 10, inertia 9639.774644478792
Iteration 11, inertia 9639.697843285941
Iteration 12, inertia 9639.504960365535
Iteration 13, inertia 9638.95769401947
Iteration 14, inertia 9638.573356540428
Iteration 15, inertia 9636.940111858781
Iteration 16, inertia 9635.980505559797
Iteration 17, inertia 9635.293549236569
Iteration 18, inertia 9633.992196284771
Iteration 19, inertia 9631.294896772604
Iteration 20, inertia 9629.157517862288
Iteration 21, inertia 9629.034229492028
Iteration 22, inertia 9628.960271079659
Iteration 23, inertia 9628.812549454027
Iteration 24, inertia 9628.623882692866
Iteration 25, inertia 9628.59305637934
Iteration 26, inertia 9628.524787673914
Iteration 27, inertia 9628.459530177464
Converged at iteration 27: strict convergence.
Initialization complete
Iteration 0, inertia 16152.65491388627
Iteration 1, inertia 10375.533418947318
Iteration 2, inertia 10144.857929200623
Iteration 3, inertia 10053.027516666378
Iteration 4, inertia 10011.007658000435
Iteration 5, inertia 9990.350407887636
Iteration 6, inertia 9980.01909365632
Iteration 7, inertia 9973.147134155095
Iteration 8, inertia 9962.93227967642
Iteration 9, inertia 9950.804233790155
```

```
Iteration 10, inertia 9929.106524971192
Iteration 11, inertia 9893.61445739094
Iteration 12, inertia 9871.997645307263
Iteration 13, inertia 9853.145440604721
Iteration 14, inertia 9818.078861860033
Iteration 15, inertia 9757.278886917615
Iteration 16, inertia 9713.510022024357
Iteration 17, inertia 9680.926171314575
Iteration 18, inertia 9674.777601266724
Iteration 19, inertia 9673.122071852886
Iteration 20, inertia 9672.685183118592
Iteration 21, inertia 9672.518137679992
Iteration 22, inertia 9672.500575556094
Converged at iteration 22: strict convergence.
Initialization complete
Iteration 0, inertia 16505.93842126068
Iteration 1, inertia 10174.540781006854
Iteration 2, inertia 9896.176365470483
Iteration 3, inertia 9755.217403382474
Iteration 4, inertia 9710.856879173405
Iteration 5, inertia 9695.291352573073
Iteration 6, inertia 9683.148926900818
Iteration 7, inertia 9667.5491075782
Iteration 8, inertia 9655.586520230188
Iteration 9, inertia 9649.747659712908
Iteration 10, inertia 9645.46312963105
Iteration 11, inertia 9644.40630542299
Iteration 12, inertia 9644.101013829977
Iteration 13, inertia 9643.642758888995
Iteration 14, inertia 9643.008226227485
Iteration 15, inertia 9642.844684153424
Iteration 16, inertia 9642.701135334233
Iteration 17, inertia 9642.57061072763
Iteration 18, inertia 9642.519519077938
Iteration 19, inertia 9642.504477800321
Converged at iteration 19: strict convergence.
Initialization complete
Iteration 0, inertia 17172.960091144152
Iteration 1, inertia 10194.671271050402
Iteration 2, inertia 10127.428286254335
Iteration 3, inertia 10110.648670434139
Iteration 4, inertia 10088.305356307605
Iteration 5, inertia 10056.98392140109
Iteration 6, inertia 10014.54147542604
Iteration 7, inertia 9984.782947200316
Iteration 8, inertia 9948.035333426422
Iteration 9, inertia 9907.185271483926
Iteration 10, inertia 9884.78043245219
Iteration 11, inertia 9863.689071376455
Iteration 12, inertia 9845.824302775172
Iteration 13, inertia 9831.477466918255
Iteration 14, inertia 9805.29705603225
Iteration 15, inertia 9788.493454238513
Iteration 16, inertia 9782.947379376854
Iteration 17, inertia 9781.810910498489
Iteration 18, inertia 9781.446129264701
Iteration 19, inertia 9781.063080143778
Iteration 20, inertia 9780.165908514351
Iteration 21, inertia 9779.376078944326
Iteration 22, inertia 9778.781046860006
Iteration 23, inertia 9777.25564144428
Iteration 24, inertia 9771.593080768085
Iteration 25, inertia 9759.993073682037
Iteration 26, inertia 9741.127172080985
Iteration 27, inertia 9725.59447725809
```

```
Iteration 28, inertia 9693.057299937595
Iteration 29, inertia 9674.633488354748
Iteration 30, inertia 9668.883857942214
Iteration 31, inertia 9668.062939210902
Iteration 32, inertia 9667.795522508974
Converged at iteration 32: strict convergence.
Initialization complete
Iteration 0, inertia 16121.151887575434
Iteration 1, inertia 10180.975017955254
Iteration 2, inertia 10012.790506392183
Iteration 3, inertia 9945.149592230093
Iteration 4, inertia 9902.19445401729
Iteration 5, inertia 9854.551471824738
Iteration 6, inertia 9796.629809393482
Iteration 7, inertia 9733.824033247713
Iteration 8, inertia 9709.60496866079
Iteration 9, inertia 9701.710992691354
Iteration 10, inertia 9698.331774194536
Iteration 11, inertia 9696.490391355273
Iteration 12, inertia 9695.20652681061
Iteration 13, inertia 9695.015903631236
Iteration 14, inertia 9694.98725574485
Converged at iteration 14: strict convergence.
Initialization complete
Iteration 0, inertia 15719.84699885392
Iteration 1, inertia 10093.886021577151
Iteration 2, inertia 9883.872126085509
Iteration 3, inertia 9824.269228240664
Iteration 4, inertia 9779.058685068434
Iteration 5, inertia 9725.75686585067
Iteration 6, inertia 9686.098533273536
Iteration 7, inertia 9677.796926090543
Iteration 8, inertia 9674.731965248277
Iteration 9, inertia 9673.142492501147
Iteration 10, inertia 9672.65337922592
Iteration 11, inertia 9672.51584737018
Iteration 12, inertia 9672.498249204844
Converged at iteration 12: strict convergence.
Initialization complete
Iteration 0, inertia 15272.42924127074
Iteration 1, inertia 10146.7544705174
Iteration 2, inertia 10060.416621065493
Iteration 3, inertia 10009.051773331206
Iteration 4, inertia 9916.53892424252
Iteration 5, inertia 9817.043864074043
Iteration 6, inertia 9800.317052776645
Iteration 7, inertia 9790.614521302858
Iteration 8, inertia 9786.58080875628
Iteration 9, inertia 9785.526700118644
Iteration 10, inertia 9783.706006793813
Iteration 11, inertia 9782.330522508842
Iteration 12, inertia 9781.663433153548
Iteration 13, inertia 9781.595470810445
Iteration 14, inertia 9781.360992187556
Iteration 15, inertia 9781.307176755947
Converged at iteration 15: strict convergence.
```

Out[61]:
```
KMeans(max_iter=500, n_clusters=3, verbose=1)
```

In [62]:
```
clusters_norm3 = kmeans3.predict(df_age_groups_norm)
```

In [63]:
```
size3 = cluster_sizes(clusters_norm3)
```

```
for c in size3.keys():
    print("Size of Cluster", c, "= ", size3[c])
```

```
Size of Cluster 0 =  541
Size of Cluster 1 =  1018
Size of Cluster 2 =  552
```

In [64]:
```
# View centroids for an aggregate representation and a characterization of each
pd.options.display.float_format='{:,.2f}'.format

centroids3 = pd.DataFrame(kmeans3.cluster_centers_, columns=df_age_groups_norm.c
centroids3
```

Out[64]:

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	family_histo
0	0.43	0.45	0.90	0.10	0.15	
1	0.48	0.27	0.39	0.61	0.26	
2	0.51	0.42	0.30	0.70	0.07	

In [65]:
```
centroids3['Age_Gen-Z'] #clusters containing Gen-Z Normalized
```

Out[65]:
```
0   0.62
1   0.99
2   0.01
Name: Age_Gen-Z, dtype: float64
```

In [66]:
```
centroids3['Age_Millenials'] #clusters containing Millenials Normalized
```

Out[66]:
```
0   0.37
1   0.01
2   0.92
Name: Age_Millenials, dtype: float64
```

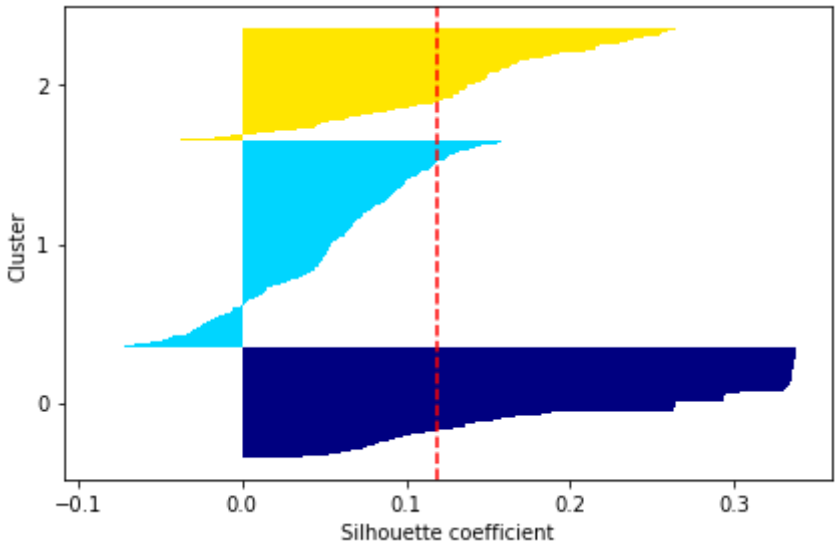
In [67]:
```
centroids3['Age_Gen-X & Boomers'] #clusters containing Gen-X and Boomers Normali
```

Out[67]:
```
0   0.01
1   0.00
2   0.07
Name: Age_Gen-X & Boomers, dtype: float64
```

In [68]:
```
# Silhouette Analysis at n = 3:
age_norm_silhouette = metrics.silhouette_samples(df_age_groups_norm, clusters_no
print('Mean Silhouette Value :', age_norm_silhouette.mean())
```

```
Mean Silhouette Value : 0.11905854619225616
```

In [69]:
```
# Plot and Evaluate the Silhouettes:
plot_silhouettes(df_age_groups_norm, clusters_norm3)
```



Above, the results are drastically different from the results from the non-normalized data. Cluster 0 outperformed all other clusters with all its coefficients above the mean silhouette value. Cluster 2 performed adequately with many of its coefficients above the mean silhouette value and only a few of its coefficients in negative. Cluster 1 did not perform as well as many of the coefficients are in negative and none of them are above the mean silhouette value. When looking at the centroids, the values of the age group do not directly correspond to the silhouette plots.

In [70]:
```
# Calculate Completeness and Homogeneity for the clusters:
complete_norm = completeness_score(labels_num, clusters_norm3)
print(f"Completeness Score for Clusters: {complete_norm}")
homogene_norm = homogeneity_score(labels_num, clusters_norm3)
print(f"Homogeneity Score for Clusters: {homogene_norm}")
```

```
Completeness Score for Clusters: 0.3552093808452009
Homogeneity Score for Clusters: 0.19224754943375816
```

These results show that with the normalized data, a pattern may not necessarily appear in the age groups. Moreover, when examining K-means and clustering, we can see how not scaling the data may lead to conclusions or patterns about the data when a pattern may not necessarily exist. This is validated when evaluating the completeness and homogeneity scores, which both resulted in low scores. The completeness score was around 0.34 and the homogeneity score is lower at 0.18. These scores show that grouping by age is not the main determining factor for the classification of obesity levels. Age still may play a role as a key feature, but the clustering exploration does not necessary reveal that the age groupings have a significant pattern. By building the classification models and performing feature selection, we will be able to obtain a better picture of age and age groupings and their role in classifying obesity levels.

Save Output of Data-Set (non-normalized) based on Age-Groups for Classifier Use:

In [71]:
```
# Create a copy of the data with the Age Groups:
data_age_groups = data_numeric
data_age_groups["Age"] = age_df['Age Group']
```

In [72]:
```
data_age_groups
```

Out[72]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no
0	Gen-Z	1.62	64.00	1	0	0
1	Gen-Z	1.52	56.00	1	0	0
2	Gen-Z	1.80	77.00	0	1	0
3	Millenials	1.80	87.00	0	1	1
4	Gen-Z	1.78	89.80	0	1	1
...	...	...	...	...	...	...
2106	Gen-Z	1.71	131.41	1	0	0
2107	Gen-Z	1.75	133.74	1	0	0
2108	Gen-Z	1.75	133.69	1	0	0
2109	Gen-Z	1.74	133.35	1	0	0
2110	Gen-Z	1.74	133.47	1	0	0

2111 rows × 43 columns

In [73]:
```
# Add the class labels as a column to the dataset:
data_age_groups['NObeyesdad'] = labels_df
data_age_groups
```

Out[73]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no
0	Gen-Z	1.62	64.00	1	0	0
1	Gen-Z	1.52	56.00	1	0	0
2	Gen-Z	1.80	77.00	0	1	0
3	Millenials	1.80	87.00	0	1	1
4	Gen-Z	1.78	89.80	0	1	1
...	...	...	...	...	...	...
2106	Gen-Z	1.71	131.41	1	0	0
2107	Gen-Z	1.75	133.74	1	0	0
2108	Gen-Z	1.75	133.69	1	0	0
2109	Gen-Z	1.74	133.35	1	0	0
2110	Gen-Z	1.74	133.47	1	0	0

2111 rows × 44 columns

In [74]:
```
genz_df = data_age_groups[data_age_groups["Age"] == 'Gen-Z']
genz_df
```

Out[74]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	fa
0	Gen-Z	1.62	64.00	1	0	0	
1	Gen-Z	1.52	56.00	1	0	0	
2	Gen-Z	1.80	77.00	0	1	0	
4	Gen-Z	1.78	89.80	0	1	1	
6	Gen-Z	1.50	55.00	1	0	0	
...	...	...	...	...	...	...	
2106	Gen-Z	1.71	131.41	1	0	0	
2107	Gen-Z	1.75	133.74	1	0	0	
2108	Gen-Z	1.75	133.69	1	0	0	
2109	Gen-Z	1.74	133.35	1	0	0	
2110	Gen-Z	1.74	133.47	1	0	0	

1353 rows × 44 columns

In [75]:
```
#Save Gen-Z dataframe to CSV:
genz_df.to_csv('/Users/cl/genz_dataframe.csv', index = False)
```

In [76]:
```
millen_df = data_age_groups[data_age_groups["Age"] == 'Millenials']
millen_df
```

Out[76]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no
3	Millenials	1.80	87.00	0	1	1
5	Millenials	1.62	53.00	0	1	1
10	Millenials	1.85	105.00	0	1	0
16	Millenials	1.93	102.00	0	1	0

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no
17	Millenials	1.53	78.00	1	0	1
...	...	...	...	...	...	...
2098	Millenials	1.61	104.95	1	0	0
2099	Millenials	1.63	108.09	1	0	0
2100	Millenials	1.63	107.38	1	0	0
2101	Millenials	1.63	107.22	1	0	0
2102	Millenials	1.63	108.11	1	0	0

717 rows × 44 columns

In [77]:
```
# Save Millenials dataframe to CSV:
millen_df.to_csv('/Users/cl/millenials_dataframe.csv', index = False)
```

In [78]:
```
genxboomers_df = data_age_groups[data_age_groups["Age"] == 'Gen-X & Boomers']
genxboomers_df
```

Out[78]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no
13	Gen-X & Boomers	1.80	99.00	0	1	1
21	Gen-X & Boomers	1.69	87.00	1	0	0
92	Gen-X & Boomers	1.78	84.00	0	1	0
133	Gen-X & Boomers	1.65	66.00	1	0	1
137	Gen-X & Boomers	1.60	80.00	0	1	0
161	Gen-X & Boomers	1.65	80.00	0	1	1
169	Gen-X & Boomers	1.63	77.00	1	0	0
197	Gen-X & Boomers	1.75	118.00	0	1	0
201	Gen-X & Boomers	1.54	80.00	1	0	0
232	Gen-X & Boomers	1.59	50.00	1	0	0

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no
252	Gen-X & Boomers	1.79	90.00	0	1	0
358	Gen-X & Boomers	1.75	110.00	0	1	0
375	Gen-X & Boomers	1.80	92.00	0	1	0
492	Gen-X & Boomers	1.70	86.00	0	1	1
751	Gen-X & Boomers	1.72	82.92	1	0	1
813	Gen-X & Boomers	1.77	75.63	1	0	0
1013	Gen-X & Boomers	1.77	80.49	0	1	1
1017	Gen-X & Boomers	1.65	79.17	1	0	0
1034	Gen-X & Boomers	1.75	82.13	0	1	0
1062	Gen-X & Boomers	1.73	86.95	1	0	0
1063	Gen-X & Boomers	1.68	79.67	1	0	0
1088	Gen-X & Boomers	1.66	80.99	0	1	0
1101	Gen-X & Boomers	1.72	88.60	0	1	0
1158	Gen-X & Boomers	1.67	80.40	0	1	0
1162	Gen-X & Boomers	1.68	79.85	1	0	0
1179	Gen-X & Boomers	1.74	84.73	0	1	0
1208	Gen-X & Boomers	1.69	80.41	1	0	0
1215	Gen-X & Boomers	1.57	81.83	1	0	0
1216	Gen-X & Boomers	1.58	81.94	1	0	0
1267	Gen-X & Boomers	1.59	76.13	1	0	0
1285	Gen-X & Boomers	1.65	86.64	1	0	0

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no
1286	Gen-X & Boomers	1.64	81.98	1	0	0
1305	Gen-X & Boomers	1.60	77.35	1	0	0
1325	Gen-X & Boomers	1.57	81.06	1	0	0
1385	Gen-X & Boomers	1.57	81.92	1	0	0
1386	Gen-X & Boomers	1.58	80.99	1	0	0
1387	Gen-X & Boomers	1.58	81.92	1	0	0
1489	Gen-X & Boomers	1.54	77.05	1	0	0
1490	Gen-X & Boomers	1.59	77.00	1	0	0
1529	Gen-X & Boomers	1.75	116.59	0	1	0
1618	Gen-X & Boomers	1.75	115.81	0	1	0

In [79]:
```
# Save Gen-X and Boomers dataframe to CSV:
genxboomers_df.to_csv('/Users/cl/genxboomers_dataframe.csv', index = False)
```

In [ ]:
```
```

Appendix C: Feature Selection with Decision Tree

Since the best classifer for the dataset was Decision Tree, we will use the Decision Tree classifier and evaluate it with the full dataset and each age-group dataset. Feature selection will be performed to obtain the best features from the classification. The top features will determine which features ultimately affect obesity levels the most. In addition, performing the classification on the different age groups will allow us to compare and contrast to see if a certain obesity level based on certain attributes effects a certain age group over another.

In [1]:
```
import numpy as np
import pylab as pl
import pandas as pd
import importlib
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import model_selection
from sklearn import tree
from sklearn import feature_selection
from sklearn import preprocessing
from sklearn import metrics
```

In [2]:
```
%pwd
```

Out[2]:
```
'/Users/cl'
```

In [3]:
```
# Load original dataset to Pandas dataframe:
df = pd.read_csv('/Users/cl/ObesityDataset.csv', header=0)
# Load transformed dataset with numeric values only:
data_numeric = pd.read_csv('/Users/cl/Obesity_numeric.csv', header=0)
# Load Gen-Z Dataframe:
genz_df = pd.read_csv('/Users/cl/genz_dataframe.csv', header=0)
# Load Millenials Dataframe:
millen_df = pd.read_csv('/Users/cl/millenials_dataframe.csv', header=0)
# Load Gen-X and Boomers Dataframe:
genxboomers_df = pd.read_csv('/Users/cl/genxboomers_dataframe.csv', header=0)
```

Decision Tree and Feature Selection with Full Dataset:

In [4]:
```
# Obtain the class label from original dataset:
labels_df =  df['NObeyesdad']
labels_df
```

Out[4]:
```
0             Normal_Weight
1             Normal_Weight
2             Normal_Weight
3        Overweight_Level_I
4       Overweight_Level_II
               ...
2106       Obesity_Type_III
2107       Obesity_Type_III
2108       Obesity_Type_III
2109       Obesity_Type_III
```

```
2110       Obesity_Type_III
Name: NObeyesdad, Length: 2111, dtype: object
```

In [5]:
```
# Transform class label into numeric:
le = preprocessing.LabelEncoder()
labels_num = le.fit_transform(labels_df)
labels_num
```

Out[5]:
```
array([1, 1, 1, ..., 4, 4, 4])
```

In [6]:
```
# View class label names and numeric association:
label_names = dict(zip(le.transform(le.classes_), le.classes_))
print(label_names)
```

```
{0: 'Insufficient_Weight', 1: 'Normal_Weight', 2: 'Obesity_Type_I', 3: 'Obesity_
Type_II', 4: 'Obesity_Type_III', 5: 'Overweight_Level_I', 6: 'Overweight_Level_I
I'}
```

In [7]:
```
# View Transformed Numeric Data:
data_numeric
```

Out[7]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_r
0	21	1.620000	64.000000	1	0	
1	21	1.520000	56.000000	1	0	
2	23	1.800000	77.000000	0	1	
3	27	1.800000	87.000000	0	1	
4	22	1.780000	89.800000	0	1	
...	...	...	...	...	...	
2106	20	1.710730	131.408528	1	0	
2107	21	1.748584	133.742943	1	0	
2108	22	1.752206	133.689352	1	0	
2109	24	1.739450	133.346641	1	0	
2110	23	1.738836	133.472641	1	0	

2111 rows × 43 columns

In [8]:
```
# Build training and test sets:
x_train, x_test, label_train, label_test = train_test_split(data_numeric, labels
```

In [9]:
```
# View Training Set:
x_train
```

Out[9]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_n
53	23	1.630000	55.000000	1	0	
267	38	1.700000	78.000000	0	1	
1825	18	1.821566	142.102468	1	0	
386	18	1.590000	53.000000	1	0	
1413	40	1.559005	77.601483	1	0	
...	...	...	...	...	...	.
960	17	1.618683	67.193585	1	0	
905	20	1.849425	85.228116	0	1	
1096	39	1.688354	79.278896	1	0	
235	19	1.690000	70.000000	1	0	
1061	23	1.725587	82.480214	0	1	

1688 rows × 43 columns

In [10]:
```
# View Testing Set:
x_test
```

Out[10]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_n
553	16	1.752755	50.000000	0	1	
331	17	1.740000	56.000000	0	1	
241	22	1.600000	66.000000	0	1	
1957	26	1.641209	111.856492	1	0	
1691	30	1.779325	120.751656	0	1	
...	...	...	...	...	...	.
1201	24	1.789193	89.393589	0	1	
363	19	1.800000	80.000000	0	1	
11	21	1.720000	80.000000	1	0	
510	22	1.675446	51.154201	1	0	
1711	28	1.758618	113.501549	0	1	

423 rows × 43 columns

In [11]:
```
# View Labels for Training Set:
label_train
```

```
array([1, 6, 4, ..., 6, 1, 6])
```

Out[11]:

In [12]:
```
# View Labels for Test Set:
label_test
```

Out[12]:
```
array([0, 1, 5, 4, 3, 4, 2, 4, 2, 2, 1, 3, 2, 2, 5, 2, 5, 3, 0, 6, 2, 2,
       6, 2, 2, 2, 2, 5, 6, 3, 2, 4, 6, 5, 0, 2, 0, 0, 6, 6, 1, 2, 5, 1,
       0, 4, 0, 0, 5, 2, 4, 2, 0, 5, 4, 2, 0, 2, 0, 1, 0, 3, 4, 6, 1, 5,
       4, 2, 6, 3, 2, 0, 4, 4, 3, 3, 0, 6, 3, 4, 5, 5, 4, 2, 0, 6, 1, 4,
       1, 4, 6, 6, 4, 5, 0, 3, 0, 5, 4, 4, 0, 2, 5, 1, 4, 6, 1, 3, 2, 6,
       2, 1, 0, 0, 6, 6, 4, 6, 0, 0, 2, 2, 2, 2, 4, 4, 5, 3, 4, 5, 1, 5,
       2, 2, 6, 2, 1, 4, 6, 3, 3, 0, 6, 6, 0, 6, 6, 5, 4, 4, 2, 2, 0, 6,
       5, 2, 4, 0, 6, 3, 2, 4, 1, 3, 4, 1, 5, 0, 6, 0, 5, 4, 5, 5, 4, 3,
       6, 3, 2, 2, 5, 5, 6, 1, 6, 3, 3, 2, 4, 3, 1, 2, 3, 1, 2, 2, 4, 2,
       0, 2, 6, 2, 5, 5, 1, 2, 0, 0, 2, 3, 6, 5, 5, 3, 4, 1, 2, 0, 1, 5,
       1, 5, 2, 5, 3, 6, 4, 4, 5, 0, 3, 5, 4, 6, 5, 1, 1, 2, 4, 3, 2, 0,
       6, 6, 3, 0, 4, 0, 5, 0, 2, 5, 6, 5, 2, 5, 6, 3, 3, 0, 3, 5, 2, 4,
       2, 4, 5, 4, 4, 4, 0, 0, 2, 3, 1, 0, 0, 1, 1, 2, 3, 5, 6, 2, 2, 1,
       6, 5, 6, 1, 0, 3, 2, 3, 3, 2, 6, 0, 0, 0, 2, 6, 6, 5, 3, 5, 1, 0,
       6, 4, 4, 5, 5, 2, 4, 5, 3, 5, 5, 3, 1, 0, 6, 6, 3, 3, 2, 1, 1, 3,
       5, 0, 5, 1, 3, 5, 4, 0, 5, 1, 1, 4, 3, 6, 6, 4, 5, 4, 6, 3, 5, 1,
       2, 6, 0, 4, 2, 6, 2, 4, 6, 0, 5, 2, 6, 5, 5, 0, 4, 4, 5, 6, 5, 3,
       0, 0, 4, 4, 1, 0, 3, 6, 4, 0, 1, 2, 3, 4, 2, 3, 2, 0, 6, 2, 2, 3,
       3, 2, 2, 2, 4, 6, 1, 0, 4, 0, 4, 6, 1, 3, 1, 1, 2, 1, 4, 2, 3, 2,
       6, 1, 6, 0, 3])
```

In [13]:
```
# Train Decision tree Classifier on the Training Data:
d_tree = tree.DecisionTreeClassifier()
dt_all = d_tree.fit(x_train, label_train)
```

In [14]:
```
# Function for Measure Performance:
def measure_performance(X, y, clf, show_accuracy=True, show_classification_repor
    y_pred = clf.predict(X)
    if show_accuracy:
         print ("Accuracy:{0:.3f}".format(metrics.accuracy_score(y, y_pred)),"\n

    if show_classification_report:
        print ("Classification report")
        print (metrics.classification_report(y, y_pred, zero_division=0),"\n")

    if show_confussion_matrix:
        print ("Confussion matrix")
        print (metrics.confusion_matrix(y, y_pred),"\n")
```

In [15]:
```
# Predict on Test Set, View Performance, and Accuracy of Decision Tree Model:
measure_performance(x_test, label_test, dt_all, show_confussion_matrix=True, sho
```

```
Accuracy:0.941

Classification report
              precision    recall  f1-score   support

           0       0.97      0.98      0.98        61
           1       0.91      0.89      0.90        45
           2       0.89      0.95      0.92        79
           3       0.95      0.96      0.95        54
           4       1.00      1.00      1.00        63
           5       0.95      0.90      0.92        61
```

```
           6       0.93      0.88      0.91        60

    accuracy                           0.94       423
   macro avg       0.94      0.94      0.94       423
weighted avg       0.94      0.94      0.94       423


Confussion matrix
[[60  1  0  0  0  0  0]
 [ 2 40  0  0  0  3  0]
 [ 0  0 75  3  0  0  1]
 [ 0  0  2 52  0  0  0]
 [ 0  0  0  0 63  0  0]
 [ 0  3  0  0  0 55  3]
 [ 0  0  7  0  0  0 53]]
```

Above, the Decision Tree classifer performed well on the full dataset, accurately classifying the classes at around 94.1% acurracy. Class 4: 'Obesity_Type_III' had a 100% accurate prediction. Class 0: 'Insufficient_Weight' and 3: 'Obesity_Type_II' achieved above 95% accuracy. Class 6: 'Obesity_Type_II' had the lowest accuracy at 91%. Below we calculate the accuracy for both the test and the training sets. The accuracy for the training set is 100% and the accuracy for the test set is 94.09%. The model is performing well and not overfitting since the accuracy for the test set is very close to the training set and not experiencing high variance.

In [16]:
```
# View the Accuracy of the Test and Training Sets:
print('Average Test Accuracy: ', d_tree.score(x_test, label_test))
print('Average Train Accuracy: ', d_tree.score(x_train, label_train))
```

```
Average Test Accuracy:  0.9408983451536643
Average Train Accuracy:  1.0
```

In [17]:
```
# Perform feature selection for top 15%
fs = feature_selection.SelectPercentile(feature_selection.chi2, percentile=15)
x_train_fs = fs.fit_transform(x_train, label_train)
```

In [18]:
```
# View the top 15% of the most important features:
print(data_numeric.columns[fs.get_support()].values)
```

```
['Age' 'Weight' 'Gender_Female' 'Gender_Male'
 'family_history_with_overweight_no' 'FCVC_Always' 'CAEC_Frequently']
```

In [19]:
```
# View scores for each top feature:
for i in range(len(data_numeric.columns.values)):
    if fs.get_support()[i]:
        print(data_numeric.columns.values[i], '\t\t\t\t', fs.scores_[i])
```

```
Age                                 470.510134679508
Weight                              11390.601482312912
Gender_Female                               274.57777589368993
Gender_Male                                 262.4874450895646
family_history_with_overweight_no                                   405.00183379903
723
FCVC_Always                                 542.9949158091111
CAEC_Frequently                                     348.88961093191773
```

In [20]:

```
# Evaluate the Classifier with the top 15% feature set:
d_tree.fit(x_train_fs, label_train)
x_test_fs = fs.transform(x_test)
measure_performance(x_test_fs, label_test, d_tree, show_confussion_matrix=True,
```

```
Accuracy:0.863

Classification report
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        61
           1       0.71      0.78      0.74        45
           2       0.84      0.84      0.84        79
           3       0.91      0.98      0.95        54
           4       1.00      1.00      1.00        63
           5       0.81      0.79      0.80        61
           6       0.78      0.72      0.75        60

    accuracy                           0.86       423
   macro avg       0.86      0.86      0.86       423
weighted avg       0.86      0.86      0.86       423


Confussion matrix
[[57  4  0  0  0  0  0]
 [ 3 35  0  0  0  6  1]
 [ 0  0 66  5  0  1  7]
 [ 0  0  1 53  0  0  0]
 [ 0  0  0  0 63  0  0]
 [ 0  9  0  0  0 48  4]
 [ 0  1 12  0  0  4 43]]
```

Above with the feature selection, using the top 15% of features, resulted in the classifier still being able to predict at an accuracy of 86.3%. Although, the accuracy reduced from the original feature set, the reduced feature set contains only seven features and still achieved a high level of accuracy. Class 1: 'Normal Weight' and Class 6: 'Overweight_level II' had the lowest accuracy score at 74% and 75% respectively. Class 4: 'Obesity Type III' achieved 100% accuracy and Class 3: 'Obesity Type II' still maintained over 95% accuracy. Moreover, for the full dataset, the top features that are associated to obesity levels is age, weight, gender, family history, FCVC and CAEC. Male and female gender as attributes are features that are salient to classifying obesity levels. This was seen during the cluster exploration which split the data into two clusters representing male and female genders. In addition to age and weight, family history, specifically with individuals indicating no history of obesity in their family is also an important feature when classifying obesity levels. This shows that hereditary, family, or environmental factors associated with families with a history of obesity, plays a role in an individuals obesity levels. Lastly, two eating habit features, always eating vegetables with meals (FCVC) and frequently eating food between meals round up the top features. Moreover, with the full dataset, physical activity features did were not included in the top 15% of features and instead, biological factors and eating habits were features that had more precedents in determining obesity levels.

Decision Tree and Feature Selection with Gen-Z Dataset:

In [21]:

```
# View Gen-Z Dataset:
genz_df
```

Out[21]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_
0	Gen-Z	1.620000	64.000000	1	0	
1	Gen-Z	1.520000	56.000000	1	0	
2	Gen-Z	1.800000	77.000000	0	1	
3	Gen-Z	1.780000	89.800000	0	1	
4	Gen-Z	1.500000	55.000000	1	0	
...	...	...	...	...	...	
1348	Gen-Z	1.710730	131.408528	1	0	
1349	Gen-Z	1.748584	133.742943	1	0	
1350	Gen-Z	1.752206	133.689352	1	0	
1351	Gen-Z	1.739450	133.346641	1	0	
1352	Gen-Z	1.738836	133.472641	1	0	

1353 rows × 44 columns

In [22]:

```
#Remove the age and class label column for Gen-Z DF:
data_genz = genz_df.iloc[:,1:43]
data_genz
```

Out[22]:

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	fa
0	1.620000	64.000000	1	0	0	
1	1.520000	56.000000	1	0	0	
2	1.800000	77.000000	0	1	0	
3	1.780000	89.800000	0	1	1	
4	1.500000	55.000000	1	0	0	
...	...	...	...	...	...	
1348	1.710730	131.408528	1	0	0	
1349	1.748584	133.742943	1	0	0	
1350	1.752206	133.689352	1	0	0	
1351	1.739450	133.346641	1	0	0	
1352	1.738836	133.472641	1	0	0	

1353 rows × 42 columns

In [23]:
```
# View Class Labels for Gen-Z DF:
labels_genz =  genz_df['NObeyesdad']
labels_genz
```

Out[23]:
```
0              Normal_Weight
1              Normal_Weight
2              Normal_Weight
3        Overweight_Level_II
4              Normal_Weight
                 ...
1348        Obesity_Type_III
1349        Obesity_Type_III
1350        Obesity_Type_III
1351        Obesity_Type_III
1352        Obesity_Type_III
Name: NObeyesdad, Length: 1353, dtype: object
```

In [24]:
```
# Transform class label into numeric:
le_z = preprocessing.LabelEncoder()
genz_labels = le_z.fit_transform(labels_genz)
genz_labels
```

Out[24]:
```
array([1, 1, 1, ..., 4, 4, 4])
```

In [25]:
```
# Build training and test sets for Gen-Z:
genz_train, genz_test, genz_label_train, genz_label_test = train_test_split(data
```

In [26]:
```
# View Gen-Z Training Set:
genz_train
```

Out[26]:

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	fa
382	1.770612	133.963349	1	0	0	
584	1.524926	42.000000	1	0	1	
6	1.780000	64.000000	0	1	0	
699	1.712061	75.000000	0	1	0	
705	1.456346	55.523481	1	0	1	
...	...	...	...	...	...	
715	1.624831	69.975607	1	0	0	
905	1.589100	72.713611	1	0	0	
1096	1.769328	105.000576	0	1	0	
235	1.600000	57.000000	1	0	1	
1061	1.607182	82.368441	1	0	0	

1082 rows × 42 columns

In [27]:
```
# View Gen-Z Testing Set:
genz_test
```

Out[27]:

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	fan
91	1.560000	51.000000	1	0	0	
442	1.759358	55.010450	1	0	0	
1078	1.738397	93.890682	1	0	0	
686	1.800000	85.000000	0	1	0	
857	1.717722	81.929910	0	1	0	
...	...	...	...	...	...	
78	1.660000	60.000000	1	0	0	
561	1.757958	52.094320	0	1	1	
292	1.700000	50.000000	1	0	1	
35	1.820000	72.000000	0	1	0	
1316	1.682594	127.427458	1	0	0	

271 rows × 42 columns

In [28]:
```
# View Gen-Z Labels for Training Set:
genz_label_train
```

Out[28]:
```
array([4, 0, 1, ..., 2, 1, 2])
```

In [29]:
```
# View Gen-Z Labels for Testing Set:
genz_label_test
```

Out[29]:
```
array([1, 0, 2, 5, 6, 1, 6, 5, 6, 0, 5, 5, 2, 1, 2, 0, 0, 2, 1, 1, 4, 3,
       5, 0, 1, 0, 1, 1, 6, 5, 4, 5, 5, 5, 2, 1, 2, 5, 5, 2, 5, 6, 1, 2,
       1, 3, 2, 6, 2, 2, 1, 3, 0, 0, 2, 6, 6, 4, 0, 5, 5, 1, 2, 2, 5, 6,
       2, 4, 0, 6, 5, 5, 4, 2, 2, 4, 4, 0, 3, 1, 0, 5, 6, 2, 1, 6, 2, 0,
       5, 0, 1, 0, 4, 5, 3, 3, 0, 5, 3, 4, 1, 0, 6, 1, 6, 3, 4, 2, 2, 2,
       1, 1, 3, 2, 0, 2, 1, 2, 4, 1, 2, 2, 5, 0, 5, 6, 5, 4, 0, 5, 0, 0,
       5, 0, 3, 2, 3, 0, 2, 5, 6, 0, 1, 6, 6, 1, 2, 4, 6, 6, 0, 2, 5, 1,
       4, 1, 0, 2, 1, 2, 3, 5, 3, 0, 1, 2, 1, 4, 2, 4, 0, 0, 2, 2, 5, 5,
       5, 1, 5, 3, 2, 0, 1, 0, 1, 2, 5, 2, 1, 6, 5, 6, 5, 1, 5, 4, 0, 2,
       1, 0, 5, 5, 5, 5, 6, 0, 2, 6, 5, 2, 5, 2, 0, 3, 5, 1, 0, 6, 0, 2,
       0, 2, 2, 5, 2, 5, 3, 0, 6, 4, 5, 5, 5, 1, 5, 4, 1, 6, 5, 4, 1, 1,
       2, 0, 3, 2, 1, 0, 5, 1, 1, 0, 6, 3, 5, 0, 5, 1, 2, 1, 0, 5, 6, 1,
       0, 0, 1, 0, 0, 1, 4])
```

In [30]:
```
# Train Decision tree Classifier on the Training Data:
dt_genz = d_tree.fit(genz_train, genz_label_train)
```

In [31]:
```
# Predict on Gen-Z Test Set, View Performance, and Accuracy of Decision Tree Mod
measure_performance(genz_test, genz_label_test, dt_genz, show_confussion_matrix=
```

```
Accuracy:0.919

Classification report
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        49
           1       0.93      0.83      0.88        48
           2       0.91      1.00      0.95        51
           3       1.00      0.72      0.84        18
           4       0.87      0.95      0.91        21
           5       0.92      0.89      0.91        55
           6       0.84      0.93      0.89        29

    accuracy                           0.92       271
   macro avg       0.92      0.90      0.91       271
weighted avg       0.92      0.92      0.92       271


Confussion matrix
[[49  0  0  0  0  0  0]
 [ 2 40  0  0  0  3  3]
 [ 0  0 51  0  0  0  0]
 [ 0  0  2 13  3  0  0]
 [ 0  0  1  0 20  0  0]
 [ 0  3  1  0  0 49  2]
 [ 0  0  1  0  0  1 27]]
```

In [ ]:
```
# View the Accuracy of the Test and Training Sets:
print('Average Test Accuracy: ', d_tree.score(genz_test, genz_label_test))
print('Average Train Accuracy: ', d_tree.score(genz_train, genz_label_train))
```

Above, the Decision Tree classifer for the Gen-Z dataset performed well with accuracy slightly lower than the full dataset at 91.9%. Class 0: 'Insufficient_Weight' and 2: 'Obesity_Type_I' achieved above 95% accuracy. Class 0: 'Insufficient_Weight' and 2: 'Obesity_Type_I' achieved above 95% accuracy. Class 3: 'Obesity_Type_II' had the lowest accuracy at 84%. Class 1: 'Normal_Weight' and 6: 'Overweight_Level_II' had the next lowest accuracy at 88% and 89% respectively. Moreover, for Gen-Z dataset, the model performed better in prediction with Class 0: 'Insufficient_Weight' and 2: 'Obesity_Type_I'. Both the full dataset and the Gen-Z dataset had lowest accuracy with Class 6: 'Overweight_Level_II'.

In [32]:
```
# Perform feature selection for top 15% of Gen-Z DF:
fs_genz = feature_selection.SelectPercentile(feature_selection.chi2, percentile=
genz_train_fs = fs_genz.fit_transform(genz_train, genz_label_train)
```

In [33]:
```
# View the top 15% of the most important features for Gen-Z:
print(data_genz.columns[fs_genz.get_support()].values)
```

```
['Weight' 'Gender_Male' 'family_history_with_overweight_no' 'FAVC_no'
 'FCVC_Always' 'NCP_2' 'CAEC_Frequently']
```

In [34]:
```
# View scores for each top feature:
for i in range(len(data_genz.columns.values)):
```

```
    if fs_genz.get_support()[i]:
        print(data_genz.columns.values[i], '\t\t\t\t', fs_genz.scores_[i])
```

```
Weight                              9715.93028639861
Gender_Male                                119.37576369900033
family_history_with_overweight_no                                  230.20882848759
723
FAVC_no                                    135.28900924705363
FCVC_Always                                292.28229522019336
NCP_2                               167.46598969723757
CAEC_Frequently                                    202.1009043591279
```

In [35]:

```
# Evaluate the Classifier with the top 15% feature set for Gen-Z DF:
d_tree.fit(genz_train_fs, genz_label_train)
genz_test_fs = fs_genz.transform(genz_test)
measure_performance(genz_test_fs, genz_label_test, d_tree, show_confussion_matri
```

```
Accuracy:0.808

Classification report
              precision    recall  f1-score   support

           0       0.91      0.88      0.90        49
           1       0.76      0.77      0.76        48
           2       0.86      0.86      0.86        51
           3       0.80      0.67      0.73        18
           4       0.95      1.00      0.98        21
           5       0.79      0.82      0.80        55
           6       0.57      0.59      0.58        29

    accuracy                           0.81       271
   macro avg       0.81      0.80      0.80       271
weighted avg       0.81      0.81      0.81       271


Confussion matrix
[[43  6  0  0  0  0  0]
 [ 4 37  0  0  0  3  4]
 [ 0  0 44  3  0  1  3]
 [ 0  0  5 12  1  0  0]
 [ 0  0  0  0 21  0  0]
 [ 0  4  0  0  0 45  6]
 [ 0  2  2  0  0  8 17]]
```

With the feature selection above, using the top 15% of features, resulted in the classifier dropping in accuracy to 80.8%. The model does not perform as well as the model using the full dataset. Class 4 had the highest accuracy at 98%, which is comparable to the full dataset which predicted class 4 at 100%. Class 6: 'Overweight_Level II' had the lowest accuracy at 58%. This shows that for the Gen-Z age group, the model is unable to classify 'Overweight_Level II' using the top 15% of features. Likely, this means that other attributes are required to accurately classify this obesity level. The model also does not classify Class 1: 'Normal_Weight' or Class 3: 'Obesity_Type_II' as well as the other classes. This is similar to the model using the full dataset which also had a lower accuracy level for Class 1: 'Normal_Weight' compared to other classes.

The top 15% of features includes weight, gender, family history with obesity, always eating vegetables with meals (FCVC), and frequently eating food between meals. These features are the same top features from the model using the full dataset except, for gender only male gender

is included. Only male gender is a top feature for the Gen-Z dataset which is interesting since both genders were included in the full dataset. Two additional eating habits features are included in the top features with the Gen-Z age group: not eating high calorie foods frequently and number of meals consumed daily. Moreover, eating habit features are the most important features in association with obesity level for Gen-Z age group along with biological and hereditary features. Similar to the model in the full dataset, physical activity features were not included in the top features for the classification of obesity levels.

Decision Tree and Feature Selection with Millenials Dataset:

In [36]:
```
# View Millenials Dataset:
millen_df
```

Out[36]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweigh
0	Millenials	1.800000	87.000000	0	1	
1	Millenials	1.620000	53.000000	0	1	
2	Millenials	1.850000	105.000000	0	1	
3	Millenials	1.930000	102.000000	0	1	
4	Millenials	1.530000	78.000000	1	0	
...	...	...	...	...	...	
712	Millenials	1.606474	104.954291	1	0	
713	Millenials	1.628855	108.090006	1	0	
714	Millenials	1.628205	107.378702	1	0	
715	Millenials	1.628470	107.218949	1	0	
716	Millenials	1.627839	108.107360	1	0	

717 rows × 44 columns

In [37]:
```
#Remove the age and class label column for Millenials DF:
data_millen = millen_df.iloc[:,1:43]
data_millen
```

Out[37]:

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	fam
0	1.800000	87.000000	0	1	1	
1	1.620000	53.000000	0	1	1	
2	1.850000	105.000000	0	1	0	
3	1.930000	102.000000	0	1	0	
4	1.530000	78.000000	1	0	1	
...	...	...	...	...	...	
712	1.606474	104.954291	1	0	0	

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	fam
713	1.628855	108.090006	1	0	0	
714	1.628205	107.378702	1	0	0	
715	1.628470	107.218949	1	0	0	
716	1.627839	108.107360	1	0	0	

717 rows × 42 columns

In [38]:

```
# View Class Labels for Millenials DF:
labels_millen =  millen_df['NObeyesdad']
labels_millen
```

Out[38]:

```
0        Overweight_Level_I
1             Normal_Weight
2            Obesity_Type_I
3       Overweight_Level_II
4            Obesity_Type_I
               ...
712        Obesity_Type_III
713        Obesity_Type_III
714        Obesity_Type_III
715        Obesity_Type_III
716        Obesity_Type_III
Name: NObeyesdad, Length: 717, dtype: object
```

In [39]:

```
# Transform class label into numeric:
le_m = preprocessing.LabelEncoder()
millen_labels = le_m.fit_transform(labels_millen)
millen_labels
```

Out[39]:

```
array([5, 1, 2, 6, 2, 6, 5, 6, 6, 3, 6, 6, 6, 3, 1, 2, 1, 1, 6, 2, 1, 1,
       1, 1, 5, 2, 1, 2, 1, 1, 5, 6, 1, 1, 5, 6, 1, 6, 3, 1, 1, 1, 5, 2,
       2, 2, 1, 5, 4, 2, 1, 1, 3, 6, 3, 6, 6, 1, 5, 0, 6, 1, 1, 1, 5, 1,
       1, 1, 1, 1, 5, 1, 5, 1, 1, 5, 1, 2, 5, 6, 5, 6, 1, 6, 1, 2, 1, 1,
       6, 4, 1, 1, 1, 1, 6, 2, 5, 2, 6, 5, 1, 1, 1, 6, 6, 1, 4, 4, 4, 4,
       0, 0, 0, 0, 0, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6,
       6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
       6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
       6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
       6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 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, 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, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
```

```
       3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4])
```

In [40]:

```
# Build training and test sets for Millenials:
mi_train, mi_test, mi_label_train, mi_label_test = train_test_split(data_millen,
```

In [41]:

```
# View Millenials Training Set:
mi_train
```

Out[41]:

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	fam
654	1.624950	111.004920	1	0	0	
544	1.836592	118.377601	0	1	0	
711	1.626580	105.037203	1	0	0	
666	1.611452	102.363149	1	0	0	
516	1.805445	119.484614	0	1	0	
...	...	...	...	...	...	
144	1.550000	62.877347	1	0	1	
645	1.607734	102.305767	1	0	0	
72	1.650000	71.000000	1	0	0	
235	1.837399	95.952027	0	1	0	
37	1.770000	85.000000	0	1	0	

573 rows × 42 columns

In [42]:

```
# View Millenials Testing Set:
mi_test
```

Out[42]:

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	fam
59	1.650000	50.000000	1	0	1	
60	1.700000	78.000000	0	1	1	
383	1.765258	114.330023	0	1	0	
159	1.680858	71.813380	1	0	0	
8	1.790000	90.000000	0	1	1	
...	...	...	...	...	...	

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	fam
635	1.654784	111.933152	1	0	0	
118	1.529834	62.903938	1	0	1	
273	1.542122	80.000000	1	0	0	
448	1.756221	119.117122	0	1	0	
34	1.550000	62.000000	1	0	0	

144 rows × 42 columns

In [43]:
```
# View Millenials Labels for Training Set:
mi_label_train
```

Out[43]:
```
array([4, 3, 4, 4, 3, 5, 1, 3, 2, 4, 4, 1, 4, 4, 6, 6, 3, 2, 3, 6, 3, 6,
       6, 3, 3, 5, 3, 3, 2, 4, 2, 6, 5, 2, 6, 2, 6, 4, 3, 6, 4, 3, 5, 2,
       4, 2, 4, 3, 3, 2, 6, 4, 1, 3, 4, 4, 3, 5, 3, 3, 2, 3, 3, 3, 2, 1,
       4, 4, 1, 3, 2, 3, 3, 3, 4, 4, 3, 4, 5, 3, 4, 4, 4, 6, 3, 5, 4, 6,
       4, 5, 4, 5, 3, 4, 4, 3, 6, 6, 2, 3, 3, 4, 2, 4, 6, 3, 3, 3, 3, 3,
       3, 2, 6, 6, 4, 3, 3, 3, 3, 3, 4, 3, 2, 5, 1, 5, 1, 2, 1, 4, 5, 3,
       6, 2, 3, 4, 2, 4, 3, 6, 4, 4, 2, 2, 4, 3, 6, 4, 5, 3, 5, 5, 4, 4,
       6, 3, 3, 2, 6, 4, 3, 3, 3, 3, 3, 5, 4, 3, 4, 3, 2, 3, 1, 3, 4, 3,
       3, 2, 3, 5, 6, 0, 3, 4, 2, 3, 5, 4, 5, 2, 4, 6, 3, 5, 6, 4, 4, 6,
       4, 2, 3, 2, 3, 6, 1, 4, 6, 6, 6, 6, 2, 6, 4, 3, 2, 2, 4, 3, 4, 3,
       1, 6, 6, 2, 2, 3, 4, 5, 3, 4, 3, 1, 4, 3, 3, 6, 6, 6, 6, 3, 4, 4,
       3, 2, 6, 5, 4, 5, 4, 4, 4, 6, 6, 3, 2, 4, 5, 3, 4, 6, 3, 4, 3, 5,
       2, 6, 5, 4, 1, 3, 2, 4, 2, 1, 6, 3, 6, 3, 4, 4, 3, 4, 1, 4, 3, 4,
       5, 3, 4, 3, 2, 3, 6, 4, 3, 3, 4, 3, 3, 6, 0, 3, 4, 2, 3, 5, 3, 1,
       4, 3, 3, 6, 2, 3, 5, 3, 6, 4, 6, 2, 3, 3, 5, 5, 4, 1, 6, 0, 2, 2,
       2, 2, 6, 6, 1, 2, 3, 4, 3, 6, 6, 5, 2, 6, 3, 3, 2, 5, 5, 3, 5, 5,
       3, 6, 6, 2, 2, 5, 5, 6, 6, 6, 6, 2, 4, 4, 4, 3, 6, 4, 4, 6, 3, 1,
       2, 3, 4, 6, 3, 4, 4, 6, 3, 3, 2, 2, 0, 5, 6, 6, 3, 1, 4, 5, 2, 6,
       1, 2, 6, 4, 4, 4, 4, 3, 3, 3, 1, 3, 3, 4, 2, 3, 6, 4, 3, 3, 4, 2,
       4, 6, 4, 2, 1, 3, 2, 6, 3, 5, 3, 6, 4, 4, 4, 3, 4, 6, 3, 4, 2, 6,
       4, 4, 3, 2, 1, 6, 3, 4, 4, 6, 2, 6, 3, 4, 2, 3, 5, 2, 4, 3, 4, 6,
       5, 3, 3, 5, 4, 6, 2, 5, 6, 4, 4, 3, 4, 3, 4, 2, 5, 2, 2, 5, 4, 3,
       4, 3, 4, 3, 6, 3, 2, 5, 2, 6, 4, 2, 6, 1, 4, 3, 2, 5, 4, 4, 5, 3,
       3, 2, 5, 1, 4, 2, 3, 3, 2, 6, 5, 2, 3, 2, 4, 2, 1, 4, 2, 3, 3, 4,
       4, 3, 3, 1, 3, 3, 4, 3, 4, 4, 2, 6, 2, 3, 2, 3, 2, 6, 3, 5, 1, 3,
       3, 6, 4, 4, 3, 4, 3, 6, 3, 3, 6, 2, 6, 2, 3, 3, 4, 5, 5, 4, 5, 6,
       6])
```

In [44]:
```
# View Millenials Labels for Testing Set:
mi_label_test
```

Out[44]:
```
array([0, 6, 3, 5, 6, 1, 2, 6, 4, 6, 3, 3, 3, 3, 2, 2, 5, 6, 4, 3, 1, 4,
       3, 3, 5, 6, 4, 2, 3, 6, 1, 1, 4, 3, 6, 3, 3, 3, 4, 3, 3, 3, 4, 3,
       3, 4, 4, 6, 6, 3, 4, 3, 3, 5, 6, 4, 2, 5, 2, 4, 3, 3, 6, 2, 2, 4,
       2, 3, 1, 3, 4, 4, 2, 1, 3, 3, 2, 2, 2, 4, 6, 6, 2, 4, 3, 1, 1, 3,
       3, 4, 4, 5, 4, 6, 1, 3, 6, 1, 3, 6, 2, 4, 6, 4, 3, 3, 2, 1, 5, 6,
       2, 4, 3, 3, 5, 6, 4, 0, 5, 3, 1, 3, 4, 4, 5, 4, 1, 3, 4, 5, 3, 3,
       4, 6, 1, 3, 1, 1, 2, 4, 5, 2, 3, 5])
```

In [45]:
```
# Train Decision tree Classifier on the Training Data:
dt_mi = d_tree.fit(mi_train, mi_label_train)
```

In [46]:
```
# Predict on Millenials Test Set, View Performance, and Accuracy of Decision Tre
measure_performance(mi_test, mi_label_test, dt_mi, show_confussion_matrix=True,
```

```
Accuracy:0.896

Classification report
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         2
           1       0.83      0.62      0.71        16
           2       0.77      0.89      0.83        19
           3       1.00      0.91      0.95        43
           4       1.00      1.00      1.00        30
           5       0.69      0.85      0.76        13
           6       0.95      0.95      0.95        21

    accuracy                           0.90       144
   macro avg       0.82      0.89      0.84       144
weighted avg       0.91      0.90      0.90       144


Confussion matrix
[[ 2  0  0  0  0  0  0]
 [ 2 10  0  0  0  3  1]
 [ 0  0 17  0  0  2  0]
 [ 0  0  4 39  0  0  0]
 [ 0  0  0  0 30  0  0]
 [ 0  1  1  0  0 11  0]
 [ 0  1  0  0  0  0 20]]
```

In [47]:
```
# View the Accuracy of the Test and Training Sets:
print('Average Test Accuracy: ', d_tree.score(mi_test, mi_label_test))
print('Average Train Accuracy: ', d_tree.score(mi_train, mi_label_train))
```

```
Average Test Accuracy:  0.8958333333333334
Average Train Accuracy:  1.0
```

Above, the Decision Tree classifer for the Millenials dataset did not perform as well as the model for the Gen-Z or the full dataset. The model achieved an accuracy of 89.6%. Similar to the two previous models, Class 4: 'Obesity_Type_III' had a prediction accuracy of 100%. Unlike the two previous models, Class 6: 'Overweight_Level II' and Class 3: 'Obesity_Type_II' performed better in this model with an accuracy of 95%. Class 0: 'Insufficient Weight' had an accuracy of 67%, which is starkly lower in accuracy compared to the previous two models! Class 1: 'Normal*Weight' also had a low accuracy at 71%. This aligns with the two previous models, which also had the lowest accuracy in predicting Class 1: 'Normal* Weight'.

In [48]:
```
# Perform feature selection for top 15% of Millenials DF:
fs_mi = feature_selection.SelectPercentile(feature_selection.chi2, percentile=15
mi_train_fs = fs_mi.fit_transform(mi_train, mi_label_train)
```

In [49]:
```
# View the top 15% of the most important features for Millenails:
print(data_millen.columns[fs_mi.get_support()].values)
```

```
['Weight' 'Gender_Female' 'Gender_Male'
```

```
'family_history_with_overweight_no' 'FCVC_Always' 'CAEC_Frequently'
'MTRANS_Automobile']
```

In [50]:
```
# View scores for each top feature:
for i in range(len(data_millen.columns.values)):
    if fs_mi.get_support()[i]:
        print(data_millen.columns.values[i], '\t\t\t\t', fs_mi.scores_[i])
```

```
Weight                              1599.4360572768592
Gender_Female                       173.42692440859898
Gender_Male                         146.10242506447887
family_history_with_overweight_no                                169.89205402997
59
FCVC_Always                         249.67035685056916
CAEC_Frequently                             166.34169934064465
MTRANS_Automobile                           110.75284152840752
```

In [51]:
```
# Evaluate the Classifier with the top 15% feature set for Millenials DF:
d_tree.fit(mi_train_fs, mi_label_train)
mi_test_fs = fs_mi.transform(mi_test)
measure_performance(mi_test_fs, mi_label_test, d_tree, show_confussion_matrix=Tr
```

```
Accuracy:0.799

Classification report
              precision    recall  f1-score   support

           0       0.40      1.00      0.57         2
           1       0.82      0.56      0.67        16
           2       0.58      0.74      0.65        19
           3       0.95      0.86      0.90        43
           4       1.00      1.00      1.00        30
           5       0.56      0.77      0.65        13
           6       0.76      0.62      0.68        21

    accuracy                           0.80       144
   macro avg       0.72      0.79      0.73       144
weighted avg       0.83      0.80      0.80       144


Confussion matrix
[[ 2  0  0  0  0  0  0]
 [ 3  9  0  0  0  3  1]
 [ 0  0 14  2  0  2  1]
 [ 0  0  4 37  0  1  1]
 [ 0  0  0  0 30  0  0]
 [ 0  1  1  0  0 10  1]
 [ 0  1  5  0  0  2 13]]
```

With the feature selection above, using the top 15% of features, resulted in the classifier dropping in accuracy to 79.9%. The model performs slightly worse than the model for Gen-Z age group. Class 4 again, had the highest accuracy at 100%, which is comparable to the full dataset which also predicted class 4 at 100%. Class 2: 'Obesity_Type_I' and Class 5: 'Overweight_Level_I' had the lowest accuracy at 65%. Class 6:'Overweight_Level_II' has a significant drop in accuracy, which prior to feature selection had a 95% prediction, and after feature selection has a 68% prediction. This shows that the features necessarily to predict Class 6 are not included in the top 15% features. The model also does not classify Class 1: 'Normal_Weight' as well as the other classes, which is consistent pattern among all the models.

In contrast, the model was able to predict Class 3: 'Obesity_Type_II' better than the model for the Gen-Z age group.

The top 15% of features includes weight, gender both male and female, family history with obesity, always eating vegetables with meals (FCVC), and frequently eating food between meals. These features are the same top features from the model using the full dataset. Unlike the previous two models, this model includes one additional top feature, a physical activity feature, means of transportation as automobile. This is interesting since previous models did not include a physical activity feature. Moreover, the model with the top 15% features for both the Millennials age group and the Gen-Z age group yielded similar accuracy for classification. The main difference is that a physical activity feature is included in the top features for Millennials which is not included for Gen-Z.

Decision Tree and Feature Selection with Gen-X & Boomers Dataset:

In [52]:
```
# View Gen-X and Boomers Dataset:
genxboomers_df
```

Out[52]:

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight
0	Gen-X & Boomers	1.800000	99.000000	0	1	
1	Gen-X & Boomers	1.690000	87.000000	1	0	
2	Gen-X & Boomers	1.780000	84.000000	0	1	
3	Gen-X & Boomers	1.650000	66.000000	1	0	
4	Gen-X & Boomers	1.600000	80.000000	0	1	
5	Gen-X & Boomers	1.650000	80.000000	0	1	
6	Gen-X & Boomers	1.630000	77.000000	1	0	
7	Gen-X & Boomers	1.750000	118.000000	0	1	
8	Gen-X & Boomers	1.540000	80.000000	1	0	
9	Gen-X & Boomers	1.590000	50.000000	1	0	
10	Gen-X & Boomers	1.790000	90.000000	0	1	
11	Gen-X & Boomers	1.750000	110.000000	0	1	
12	Gen-X & Boomers	1.800000	92.000000	0	1	
13	Gen-X & Boomers	1.700000	86.000000	0	1	

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight
14	Gen-X & Boomers	1.721854	82.919584	1	0	
15	Gen-X & Boomers	1.768231	75.629310	1	0	
16	Gen-X & Boomers	1.769269	80.491339	0	1	
17	Gen-X & Boomers	1.647768	79.165306	1	0	
18	Gen-X & Boomers	1.745528	82.130728	0	1	
19	Gen-X & Boomers	1.733875	86.945380	1	0	
20	Gen-X & Boomers	1.675953	79.668320	1	0	
21	Gen-X & Boomers	1.657221	80.993213	0	1	
22	Gen-X & Boomers	1.718097	88.600878	0	1	
23	Gen-X & Boomers	1.673394	80.400306	0	1	
24	Gen-X & Boomers	1.678610	79.849252	1	0	
25	Gen-X & Boomers	1.743935	84.729197	0	1	
26	Gen-X & Boomers	1.687326	80.413997	1	0	
27	Gen-X & Boomers	1.569234	81.827288	1	0	
28	Gen-X & Boomers	1.583943	81.936398	1	0	
29	Gen-X & Boomers	1.587546	76.126112	1	0	
30	Gen-X & Boomers	1.646390	86.639861	1	0	
31	Gen-X & Boomers	1.643786	81.978743	1	0	
32	Gen-X & Boomers	1.595165	77.354744	1	0	
33	Gen-X & Boomers	1.567973	81.056851	1	0	
34	Gen-X & Boomers	1.571417	81.918809	1	0	
35	Gen-X & Boomers	1.584322	80.986496	1	0	

	Age	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight
36	Gen-X & Boomers	1.582523	81.919454	1	0	
37	Gen-X & Boomers	1.544937	77.053948	1	0	
38	Gen-X & Boomers	1.592316	77.001030	1	0	
39	Gen-X & Boomers	1.750000	116.594351	0	1	
40	Gen-X & Boomers	1.750000	115.806977	0	1	

41 rows × 44 columns

In [53]:
```
#Remove the age and class label column for Millenials DF:
data_genxb = genxboomers_df.iloc[:,1:43]
data_genxb
```

Out[53]:

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	famil
0	1.800000	99.000000	0	1	1	
1	1.690000	87.000000	1	0	0	
2	1.780000	84.000000	0	1	0	
3	1.650000	66.000000	1	0	1	
4	1.600000	80.000000	0	1	0	
5	1.650000	80.000000	0	1	1	
6	1.630000	77.000000	1	0	0	
7	1.750000	118.000000	0	1	0	
8	1.540000	80.000000	1	0	0	
9	1.590000	50.000000	1	0	0	
10	1.790000	90.000000	0	1	0	
11	1.750000	110.000000	0	1	0	
12	1.800000	92.000000	0	1	0	
13	1.700000	86.000000	0	1	1	
14	1.721854	82.919584	1	0	1	
15	1.768231	75.629310	1	0	0	
16	1.769269	80.491339	0	1	1	
17	1.647768	79.165306	1	0	0	
18	1.745528	82.130728	0	1	0	
19	1.733875	86.945380	1	0	0	

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	famil
20	1.675953	79.668320	1	0	0	
21	1.657221	80.993213	0	1	0	
22	1.718097	88.600878	0	1	0	
23	1.673394	80.400306	0	1	0	
24	1.678610	79.849252	1	0	0	
25	1.743935	84.729197	0	1	0	
26	1.687326	80.413997	1	0	0	
27	1.569234	81.827288	1	0	0	
28	1.583943	81.936398	1	0	0	
29	1.587546	76.126112	1	0	0	
30	1.646390	86.639861	1	0	0	
31	1.643786	81.978743	1	0	0	
32	1.595165	77.354744	1	0	0	
33	1.567973	81.056851	1	0	0	
34	1.571417	81.918809	1	0	0	
35	1.584322	80.986496	1	0	0	
36	1.582523	81.919454	1	0	0	
37	1.544937	77.053948	1	0	0	
38	1.592316	77.001030	1	0	0	
39	1.750000	116.594351	0	1	0	
40	1.750000	115.806977	0	1	0	

41 rows × 42 columns

In [54]:
```
# View Class Labels for Gen-X & Boomers DF:
labels_genxb =  genxboomers_df['NObeyesdad']
labels_genxb
```

Out[54]:
```
0          Obesity_Type_I
1          Obesity_Type_I
2       Overweight_Level_I
3            Normal_Weight
4          Obesity_Type_I
5      Overweight_Level_II
6      Overweight_Level_II
7         Obesity_Type_II
8          Obesity_Type_I
9            Normal_Weight
10     Overweight_Level_II
11        Obesity_Type_II
12     Overweight_Level_II
13     Overweight_Level_II
```

```
14     Overweight_Level_I
15     Overweight_Level_I
16    Overweight_Level_II
17    Overweight_Level_II
18    Overweight_Level_II
19    Overweight_Level_II
20    Overweight_Level_II
21    Overweight_Level_II
22    Overweight_Level_II
23    Overweight_Level_II
24    Overweight_Level_II
25    Overweight_Level_II
26    Overweight_Level_II
27         Obesity_Type_I
28         Obesity_Type_I
29         Obesity_Type_I
30         Obesity_Type_I
31         Obesity_Type_I
32         Obesity_Type_I
33         Obesity_Type_I
34         Obesity_Type_I
35         Obesity_Type_I
36         Obesity_Type_I
37         Obesity_Type_I
38         Obesity_Type_I
39        Obesity_Type_II
40        Obesity_Type_II
Name: NObeyesdad, dtype: object
```

In [55]:
```
# Transform class label into numeric:
le_x = preprocessing.LabelEncoder()
genxb_labels = le_m.fit_transform(labels_genxb)
genxb_labels
```

Out[55]:
```
array([1, 1, 3, 0, 1, 4, 4, 2, 1, 0, 4, 2, 4, 4, 3, 3, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2])
```

In [56]:
```
# Build training and test sets for Gen-X and Boomers:
xb_train, xb_test, xb_label_train, xb_label_test = train_test_split(data_genxb,
```

In [57]:
```
# View Gen-X and Boomers Training Set:
xb_train
```

Out[57]:

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	famil
22	1.718097	88.600878	0	1	0	
21	1.657221	80.993213	0	1	0	
32	1.595165	77.354744	1	0	0	
27	1.569234	81.827288	1	0	0	
33	1.567973	81.056851	1	0	0	
29	1.587546	76.126112	1	0	0	
31	1.643786	81.978743	1	0	0	

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	famil
40	1.750000	115.806977	0	1	0	
4	1.600000	80.000000	0	1	0	
14	1.721854	82.919584	1	0	1	
10	1.790000	90.000000	0	1	0	
36	1.582523	81.919454	1	0	0	
24	1.678610	79.849252	1	0	0	
26	1.687326	80.413997	1	0	0	
35	1.584322	80.986496	1	0	0	
20	1.675953	79.668320	1	0	0	
18	1.745528	82.130728	0	1	0	
25	1.743935	84.729197	0	1	0	
6	1.630000	77.000000	1	0	0	
13	1.700000	86.000000	0	1	1	
7	1.750000	118.000000	0	1	0	
39	1.750000	116.594351	0	1	0	
1	1.690000	87.000000	1	0	0	
16	1.769269	80.491339	0	1	1	
0	1.800000	99.000000	0	1	1	
15	1.768231	75.629310	1	0	0	
5	1.650000	80.000000	0	1	1	
11	1.750000	110.000000	0	1	0	
9	1.590000	50.000000	1	0	0	
8	1.540000	80.000000	1	0	0	
12	1.800000	92.000000	0	1	0	
37	1.544937	77.053948	1	0	0	

32 rows × 42 columns

In [58]:
```
# View Gen-X and Boomers Testing Set:
xb_test
```

Out[58]:

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	famil
3	1.650000	66.000000	1	0	1	
2	1.780000	84.000000	0	1	0	

	Height	Weight	Gender_Female	Gender_Male	family_history_with_overweight_no	famil
23	1.673394	80.400306	0	1	0	
38	1.592316	77.001030	1	0	0	
17	1.647768	79.165306	1	0	0	
28	1.583943	81.936398	1	0	0	
19	1.733875	86.945380	1	0	0	
34	1.571417	81.918809	1	0	0	
30	1.646390	86.639861	1	0	0	

9 rows × 42 columns

In [59]:
```
# View Gen-X and Boomers Labels for Training Set:
xb_label_train
```

Out[59]:
```
array([4, 4, 1, 1, 1, 1, 1, 2, 1, 3, 4, 1, 4, 4, 1, 4, 4, 4, 4, 4, 2, 2,
       1, 4, 1, 3, 4, 2, 0, 1, 4, 1])
```

In [60]:
```
# View Gen-X and Boomers Labels for Testing Set:
xb_label_test
```

Out[60]:
```
array([0, 3, 4, 1, 4, 1, 4, 1, 1])
```

In [61]:
```
# Train Decision tree Classifier on the Training Data:
dt_xb = d_tree.fit(xb_train, xb_label_train)
```

In [62]:
```
# Predict on Gen-X and Boomers Test Set, View Performance, and Accuracy of Decis
measure_performance(xb_test, xb_label_test, dt_xb, show_confussion_matrix=True,
```

```
Accuracy:0.667

Classification report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.80      1.00      0.89         4
           3       0.00      0.00      0.00         1
           4       0.50      0.67      0.57         3

    accuracy                           0.67         9
   macro avg       0.33      0.42      0.37         9
weighted avg       0.52      0.67      0.59         9


Confussion matrix
[[0 0 0 1]
 [0 4 0 0]
 [0 0 0 1]
 [0 1 0 2]]
```

In [63]:
```
# View the Accuracy of the Test and Training Sets:
print('Average Test Accuracy: ', d_tree.score(xb_test, xb_label_test))
print('Average Train Accuracy: ', d_tree.score(xb_train, xb_label_train))
```

```
Average Test Accuracy:  0.6666666666666666
Average Train Accuracy:  1.0
```

Above, the Decision Tree classifer for the Gen-X and Boomers dataset performed the worse compared to all previous models. The model achieved an accuracy of 66.7%. This model resulted in the lowest accuracy score compared to the previous models. This dataset is significantly smaller than the previous two dataset. As such, not all classes are represented in this model and due to the limited number of entries, the model does not have as much data for the classifier to train on compared to previous three models. This model was able to predict Class 1: 'Normal_weight' at 89% accuracy, which is higher in accuracy compared to all previous models. This model was unable to predict Class 0: 'Insufficient_Weight or Class 3: 'Obesity_Type_II'.

In [64]:
```
# Perform feature selection for top 15% of Gen-X and Boomers DF:
fs_xb = feature_selection.SelectPercentile(feature_selection.chi2, percentile=15
xb_train_fs = fs_xb.fit_transform(xb_train, xb_label_train)
```

In [65]:
```
# View the top 15% of the most important features for Millenails:
print(data_genxb.columns[fs_xb.get_support()].values)
```

```
['Weight' 'FCVC_Always' 'CH2O_More than 2 L' 'SCC_yes' 'FAF_2 or 4 days'
 'FAF_4 or 5 days' 'MTRANS_Public_Transportation']
```

In [66]:
```
# View scores for each top feature:
for i in range(len(data_genxb.columns.values)):
    if fs_xb.get_support()[i]:
        print(data_genxb.columns.values[i], '\t\t\t\t', fs_xb.scores_[i])
```

```
Weight                          59.13977455691617
FCVC_Always                             9.282051282051283
CH2O_More than 2 L                              9.376068376068377
SCC_yes                                 31.0
FAF_2 or 4 days                         15.333333333333334
FAF_4 or 5 days                         15.0
MTRANS_Public_Transportation                    31.0
```

In [67]:
```
# Evaluate the Classifier with the top 15% feature set for Gen-X and Boomers DF:
d_tree.fit(xb_train_fs, xb_label_train)
xb_test_fs = fs_xb.transform(xb_test)
measure_performance(xb_test_fs, xb_label_test, d_tree, show_confussion_matrix=Tr
```

```
Accuracy:0.667

Classification report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       1.00      0.50      0.67         4
           3       0.50      1.00      0.67         1
           4       0.60      1.00      0.75         3
```

```
    accuracy                           0.67         9
   macro avg       0.53      0.62      0.52         9
weighted avg       0.70      0.67      0.62         9


Confussion matrix
[[0 0 1 0]
 [0 2 0 2]
 [0 0 1 0]
 [0 0 0 3]]
```

With the feature selection above, using the top 15% of features, resulted in the classifier dropping in accuracy to 66.7%. This model underperformed compared to all previous models with all classes having accuracy scores of 75% or lower. Again, the model was unable to predict Class 0: 'Insufficient_Weight.' Since some classes are not represented in this dataset and with a lower amount of data for training, it is not unexpected that the model was unable to classify obesity levels as well as the previous models.

The top 15% of features includes weight and always eating vegetables with meals (FCVC) which are two features also included as top features for the full dataset, Gen-Z dataset, and Millennial's dataset. Additional eating habits features are included as top features: water intake at more than 2 liters per day and monitoring calories intake daily. In addition, physical activity features include direct physical activity 1 to 2 days or 3 to 4 days and means of transportation by public transit. This is interesting since previous models did not include specific eating habit features such as water intake and direct exercise or direct physical activity. The results are drastically different from the Gen-Z and Gen-X dataset but since the sample size is significantly lower, more data would be needed for this population to perform a more detailed and thorough analysis in validating these top features and determining what key features affect the classification of obesity for the Gen-X and Boomers age group

Comparsion of Results:

The model with the best accuracy from the Decision Tree classifier is the full dataset. The top 15% features for this model include age, weight, gender, family history with obesity, always eating vegetables with meals (FCVC) and frequently eating food between meals (CAEC). With these top features, the model still performed well with an accuracy of 86.3%. Biological features and family history with obesity are top features that are associated with classifying obesity. With the full dataset, only two additional eating habit features were top features. The models for Gen-Z age group and Millenials age group also included weight, gender, family history with obesity, always eating vegetables with meals (FCVC), and frequently eating food between meals (CAEC) as top features. Gen-Z includes more eating habit features including not eating high calorie foods frequently and number of meals consumed daily and Millenials includes a physical activity feature which is transportation by automobile. The accuracy for the the Gen-Z model is 91.8% whereas the accuracy for the Millenails model is 89.5%. When evaluating the performance with th top 15% features, the Millenials model performs slighly better at 80.6% wheras the Gen-Z model had an accuracy of 80.1%. The model for the Gen-X and Boomers age group performed the worst at an accuracy of 77.8%. Gen-X and Boomers had different top features compared to all other models. The top features still included weight, but no longer included gender and

instead includes both eating habits and physical activity features including water intake of 2 liters or more, calories intake daily, and direct physical activity. The model had significantly lower amount of data compared to previous models which may contribute to the lower accuracy and the low performance of the model.

In conclusion, Gen-Z age group is over represented in the full datset compared to Millenials and Gen-X and Boomers. The classifier model performed better on the full dataset. The classifer model equally performed well on the Gen-Z and Millenials datset. By looking at the top features and evaluating the models using the top features, we can see which features are most important in classifying obesity levels. In this case, for the Gen-Z and Millenials age group, biological and hirediary features are more associated with obesity levels, with eating habits as additional top features specifically eating vegetables with meals and eating between meals. Gender appears to play a role with Gen-Z and Millenials age group. This is expected due to biological factors such as difference in weight, height, and calorie intake. For Gen-X and Boomers age group, weight and direct eating habits such as water intake, calorie intake, direct physical activity, and mode of transportation are top features. More data is needed for Gen-X and Boomers to be able to analysis and evaluate the models and determine which features affect classification of obesity levels best.

In [ ]: