Instituto Politécnico de Setúbal

Escola Superior de Tecnologia do Barreiro

Projeto de Big Data

Licenciatura em Bioinformática

Predict obesity using PySpark

January, 2023

Group
Matilde Machado (202000174)
Rodrigo Pinto (202000177)

Contents

1 Introduction

Obesity is a major public health concern that has been on the rise in recent years. It is a leading cause of various chronic diseases such as diabetes and heart disease. The ability to accurately predict obesity levels in a population is essential for addressing this issue and identifying high-risk individuals and communities. This report aims to demonstrate the use of PySpark, a powerful tool for big data processing, in predicting obesity levels using a dataset of lifestyle information.

The main objective of this report is to use PySpark to build a predictive model that can accurately predict obesity levels in a population based on lifestyle information. The report will cover the process of data cleaning, feature engineering, model selection and evaluation, and demonstrate the potential of PySpark in handling large datasets for predictive modeling.

Machine learning is a rapidly growing field of computer science that aims to develop algorithms and statistical models that enable computers to learn from data, without being explicitly programmed. Machine learning has become increasingly important in recent years due to the abundance of data being generated and the need for more efficient and effective ways to analyze it.

There are several types of machine learning, each with its own set of characteristics and applications. Supervised learning is the most common form of machine learning, where the algorithm is trained on a labeled dataset, and is then used to predict the outcome for new, unseen data. Unsupervised learning, on the other hand, is used when the data is not labeled, and the algorithm is used to discover patterns and structure in the data. Reinforcement learning is a type of machine learning that focuses on learning from experience, by taking actions in an environment to maximize a reward. This type of learning is commonly used in robotics, gaming and autonomous vehicles.

One of the most significant benefits of machine learning is its ability to automatically identify patterns and insights in data that would be difficult or impossible for humans to detect. This makes it an ideal tool for a wide range of applications, such as natural language processing, image and speech recognition, and predictive modeling. In the healthcare industry, machine learning is being used to predict disease outcomes and identify high-risk patients. Another important aspect to consider is the ethical implications of machine learning. As the technology evolves, it is important to ensure that it is being used in a fair and responsible manner, and that the privacy and security of individuals is protected.

PySpark is an open-source Python library for big data processing that allows for easy and efficient data analysis on large-scale datasets. It is built on top of the Apache Spark framework, which is a fast and general-purpose cluster computing system. PySpark pro-

vides a simple and easy-to-use API for data scientists and engineers to interact with large-scale data in a distributed environment.
One of the main advantages of PySpark is its ability to handle large-scale data processing with ease. It is designed to work with large datasets that are distributed across multiple machines, allowing for faster and more efficient data analysis. The library also supports a variety of data formats, including text files, JSON, and Parquet, making it a versatile tool for data processing.
PySpark also provides a wide range of functionality for data processing and analysis. It includes functionality for data cleaning, feature engineering, and machine learning, making it a powerful tool for data scientists and engineers. The library also has a number of built-in machine learning libraries, such as MLlib and GraphX, which can be used for tasks such as classification, regression, and clustering. With its built-in machine learning libraries, PySpark can also help data scientists to perform advanced data analysis and modeling tasks. The library has been widely adopted by industry and academia and it's been gaining more popularity and support by the community.

2 Sections

2.1 Collecting Data

As bioinformatics students, we were interested in studying a specific disease. After researching various datasets available, we decided to use the "Estimation of obesity levels based on eating habits and physical condition Data Set"[1] dataset for our analysis.

2.2 Dataset Description

The dataset being described contains information on various characteristics of population form Colombia, Peru and Mexico. The data was collected from a anonymous web survey with 16 questions ranging from age and gender to food and health habits. It includes demographic information such as age, gender, as well as information on behaviours and preferences, for example water consumption, smoking, caloric food, etc. The dataset is intended to be used for analysis to identify trends and understand how different factors affect obesity.
The Gender variable can have 2 values: female, male.
The Age variable is a numeric value between 14 and 61.
The Height variable and the Weight variable are both numeric values.
The family_history_with_overweight variable, is if a person has any family member with or who had overweight, it has 2 values: yes, no.

The FAVC variable, is how frequently a person has caloric food a day, it has 2 values: yes, no.
The FCVC in meals variable, is how frequently a person has vegetables in their meals, it has 3 values: never, sometimes, always.
The NCP variable, is the amount of main meals a person has a day, it can have 3 values: between 1 and 2, three, more than three.
The CAEC, is if the person eats between meals, it has 4 values: no, sometimes, frequently, always.
The SMOKE variable, is if a person smokes, it has 2 values: yes, no.
The CH2O variable, is how much water a person consumes daily, it can have 3 values: less than a litter, between 1 and 2 litter, more than 2 litter.
The SCC variable, is if a person monitors their daily calories, it can have 2 values: yes, no.
The FAF variable, is how often a person exercises, it can have 4 values: i do not have, 1 or 2 days, 2 or 4 days, 4 or 5 days.
The TUE variable, is how much time a person spends on technological devices, it can have 3 values: 0-2 hours, 3-5 hours, more than 5 hours.
The CALC variable, is how often a person drinks alcohol, it can have 4 values: i do not drink, sometimes, frequently, always.
The MTRANS variable, is the transportation mean that a person usually uses, it can have 5 values: Automobile, Motorbike, Bike, Public Transportation, Walking.
The NObeyesdad variable, was created by the authors based on BMI (body mass index), using the formula and a key:
Underweight Less than 18.5
Normal 18.5 to 24.9
Overweight 25.0 to 29.9
Obesity I 30.0 to 34.9
Obesity II 35.0 to 39.9
Obesity III Higher than 40
This dataset has 23% of data etrived form the survey, the rest of the data was generated syntheticly, because the data was unbalnced. From the survey were retrived 485 records and almost 300 of them were normal weigth, the rest of the catgories of the NObeyesdad variable were not significant, this resulted in havig almost perfect predictions of the normal weight category and not being able to predict the other categories. So the authores used SMOTE to generate the rest of the data.

2.3 Data Processing

Before starting the exploratory analysis, it was necessary to change somethings about the dataset. The first thing we had to do was to change the categorical columns to numerical because pyspark AssembleVector only works with numerical columns and to accomplish that we used the "withColumn" and "cast" command. The only columns that we didn't change were the numerical columns and the column of obesity because every model will use that column in a different way and we will explain that later on this report. After having almost all columns with numerical type, it was time to change the names of our columns to make the dataset cleaner. In the end we also count the number of nulls per column and found out that our dataset had no nulls.

```
df = df.withColumn("Gender", when(col("Gender")== "Female", 0).
   otherwise(1))
df = df.withColumnRenamed('family_history_with_overweight', 'FH')
```

To confirm that everythings was good to go, we observed the schema of our dataset and we found out that the "numerical" columns were actually strings type columns, so it was needed to change those columns to the integer type for that we used the cast command as it shows in the next script.

```
df = df.withColumn("VEGET",col("VEGET").cast(IntegerType()))
```

2.4 Exploratory Data Analysis

It was performed an exploratory analysis to better understand our dataset.

- Number of rows in the dataframe: 2111
- Which type of obesity level is most common: Obesity Type 1
- How many people have Obesity Type I: 351
- Which type of obesity level is least common: Insufficient Weight
- How many people are diagnosed with obesity: 972

2.5 Logistic Regression Model

It was decided that the first model would be the Logistic Regression Classification Model. Logistic Regression is a supervised machine learning algorithm that is used for classification problems. It is a type of generalized linear model that is used to model the probability of a binary outcome, given a set of input features. The algorithm uses the logistic function, which maps the input features to a probability between 0 and 1. The output of the logistic function is then thresholded to produce a binary output. The main advantage of logistic regression is that it is easy to interpret, and it provides a probability score for the classification.

Our objective with this model is to predict with the given features if a person is or is not obese. For that purpose we will change the obesity column in a way that will only have 2 categories, 0 for the people who are not obese (insufficient weight, normal weight and overweight I and II) and one for the people are obese (Obesity type I II and III). Afterwards we combine the features with the column obesity in a new dataset. The next line of code will do exactly that.

```
df = df.withColumn("OBESITY", when(col("OBESITY")== "Insufficient
   _Weight", 0).when(col("OBESITY") == "Normal_Weight", 0).when(
   col("OBESITY")=="Overweight_Level_I", 0).when(col("OBESITY")==
    "Overweight_Level_II",0).otherwise(1))
```

Now, we have the dataset completely ready for starting the model, the first thing we have to do is the Vector that will contain any feature that we want to use. After having that vector we will standardized that vector. Standardization is a technique used to transform data into a common scale. The process of standardization involves subtracting the mean from each data point and dividing by the standard deviation. The next excerpt shows how we did this 2 steps

```
assemble=VectorAssembler(inputCols=[
    'FH','CALORICFOOD', "SMOKE", "VEGET", "MEALS",
    "BETWEENMEALS", "H2O", "CALORIES", "PHYSY", "ALCH", "WEIGHT"
], outputCol='features')
assembled_data=assemble.transform(df)
scale=StandardScaler(inputCol='features',outputCol='standardized'
   )
scale=scale.fit(assembled_data)
scale_output= scale.transform(assembled_data)
scale_output.select('standardized').show(2,truncate=False)
```

After having the data standardized, it was time to split the data and build the model with the train data set and the predictions model with the test dataset. We chose to split in 80/20 because we had a lot of features so we needed our train dataset to be this big. Now we have to analyse the model, to do that we used the BinaryClassificationEvaluator from pyspark to evaluate the model and we found out that our model had the great evaluation of 0.90 and that means that our score has an excellent discrimination. To vizualize how good was our model we decdied to build to things. Using sklearn library we built a confusion matrix and a roc curve and the results are found below.

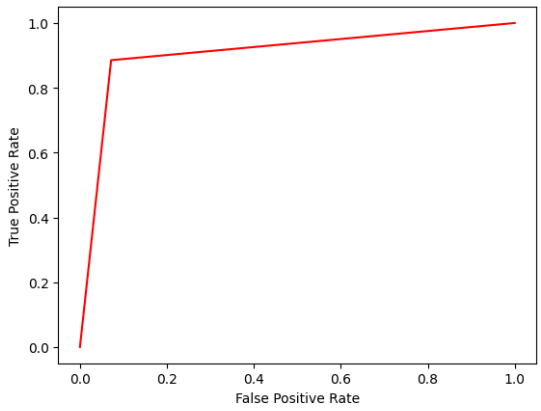

Figure 1: ROC Curve

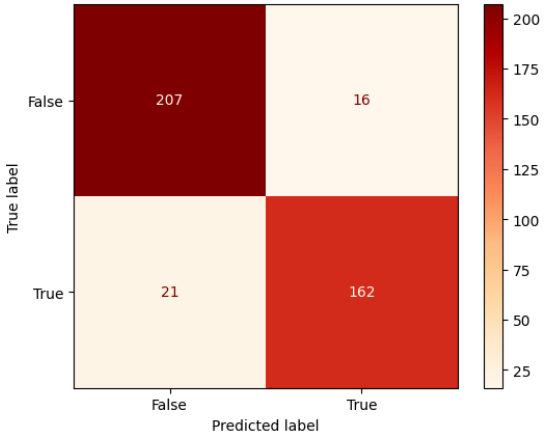

Figure 2: Confusion Matrix

2.5.1 Results

We can observe that the model have a great percentage of True positives and True Negatives and not so much of the false positives and negatives, and our ROC curve looks how it should be with a 0.90 AUC (Area under the curve). We also calculate the accuracy of our model and it was close to 0.90.

To finish this model, we tried to improve the accuracy of the model using ParamGridBuilder and Crossvalidator commands from pyspark and we only got to improve the accuracy by 0.1.

2.6 K-Means Clustering

K-means clustering is a popular unsupervised machine learning technique that is used to group similar data points together. It is a type of partitioning clustering algorithm, which divides the dataset into k clusters, where k is a user-specified parameter. The goal of k-means is to minimize the sum of squared distances between the data points and their closest cluster center, also known as the centroid. Before implementing the Kmeans algorithm we need to standarize and scale the data, for that we used VectorAssembler and StandardScaler. We do this because K-means is sensitive to the scale of the variables, this means that a variable that has a larger scale will dominate the distance calculations, making it dificukt for the algorithm to identify clusters in the data. The process of transforming the data to have a mean of zero and a standard deviation of one, is done by subtracting the mean from each value and dividing by the standart deviation. By standarizing the data, all variables have an equal impact on the distance calculations. This allows the K-means algorithm to identify the clusters based on the true underlying structure of the data, rather than being influenced by the scale of the variables.
To determine the optimal number of clusters to use in the K-means algorithm we use the Elbow Method. To do this we run the K-means algorithm for a range of diferent values of k an then plot the sum of squared distances of the points within each cluster against the number of clusters. The number of clusters is chosen as the value of k where the change in the sum of the squared distances begins to level off and becomes relatively small. From the plot we gather that the optimal number of clusters is 7.
We implemented the K-means algorithm for 7 clusters.

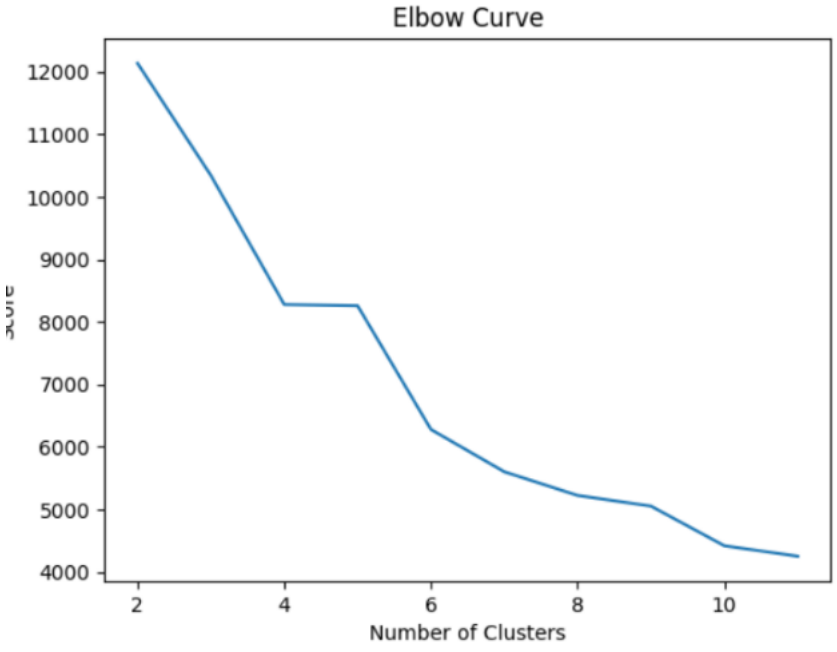

Figure 3: Elbow Curve

To vizualize the clusters we used PCA, which is Principal component analysis, a dimensionality reduction thecnique that is used to project high-dimensional data onto a lower-dimensional space, while preserving as much of the original variance in the data as possible. in this case is used to reduce the dimensionality to 2 or 3 dimensions so that it can be plotted and vizualized, this allows for the clusters identified by the K-means algorithm to be more easily interperted and understood. In a 2 dimensional plot, the first principal component is plotted on the x-axis, and the second principal component is plotted in the y-axis. Each point represents a datapoint, and is colored according to the cluster it belogs to. This allowed us to visualize the clusters in a way that it is easier to interperter and understand.

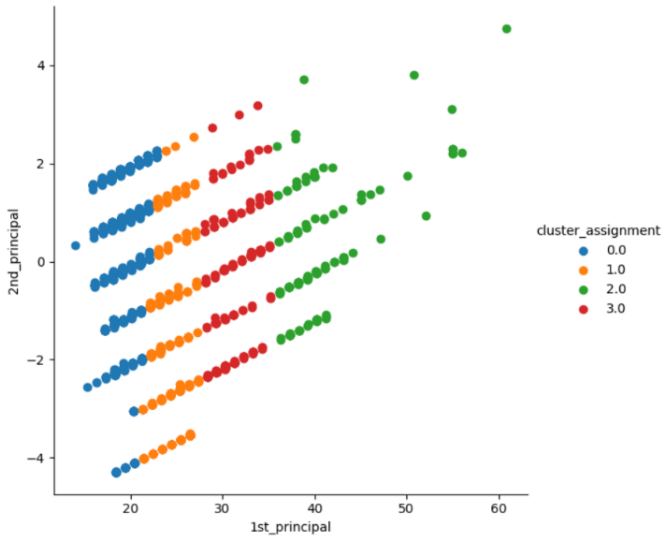

Figure 4: PCA

2.6.1 Results

To analize the clusters we used the Silhouette score, which ranges from -1 to 1, where a score pf 1 indicates a perfect clustering solution and a score of -1 indicates a poor clustering solution, a score close to 0 suggest that the clusters are not well-defined and the objects are not well separated from the other clusters. Our score is 0.07 , which indicates that the objects in the clusters are only slightly similar to the objects in their own clusters compared to the objects in other clusters. This could be an indication of poor clustering or a poor choice of the number of clusters. It is important to check the data and the clustering method used, it could be that the data is not well suited for the clustering algorithm used or that the parameters of the algorithm are not well-chosen. Also, it could be that the chosen number of clusters is not appropriate for the data and it should be adjusted.

2.7 Decision Tree

Decision Trees are a type of supervised machine learning algorithm that is used for both classification and regression problems. The algorithm works by recursively splitting the dataset into smaller subsets, based on the values of the input features. Each split is made based on the feature that maximizes the reduction in impurity, such as Gini impurity or entropy. The final output of the algorithm is a tree-like structure, where each leaf node represents a class or a value, and each internal node represents a test on a feature. Decision Trees are easy to interpret and can handle non-linear relationships between the input features and the output.

In the first model where we used the logistic regression we were trying to predict if a person was or not obese. Now with the help of the decision tree classifier model we will try to go more deep into the concept and try to predict the weight category of a person. Remember that our data set had 6 weight categories in the obesity column that were "insufficient_weight", "normal_weight", "Overweight I and II" and "Obesity I, II and III". Like the logistic regression model, the first thing we had to do was to change the obesity column to numerical, but this time with multiple numbers

```
df_knn = df_knn.withColumn("NObeyesdad", when(col("NObeyesdad")==
    "Insufficient_Weight", 1).when(col("NObeyesdad") == "Normal_
   Weight", 2).when(col("NObeyesdad")=="Overweight_Level_I", 3).
   when(col("NObeyesdad")== "Overweight_Level_II",4).when(col("
   NObeyesdad")=="Obesity_Type_I", 5).otherwise(6))
```

For the assemble it was decided to add more columns because this model needs to be more accurate. We add the age and gender columns to get a at least good accuracy. Afterwards we did the standardization and the split exactly like in the Logistic Regression model. For the decision tree we used the DecisionTreeClassifier from pyspark and the MulticlassClassification model also from pyspark to evaluate the model. We used a max depth of 4 so that our decision tree wouldnt be that big and impossible to analyse. In the end we got a auc score of 0.78 which is really good for a decision tree.

```
deciTree = DecisionTreeClassifier(featuresCol="standardized",
   labelCol="OBESITY", maxDepth=4)
predTree = deciTree.transform(testT)
evaluator=MulticlassClassificationEvaluator(predictionCol="
   prediction", labelCol="OBESITY")
acc = evaluator.evaluate(predTree)
print(acc)
```

Before visualizing the tree, we had to build a confusion matrix as we did in logistic regression to analyse better our model.

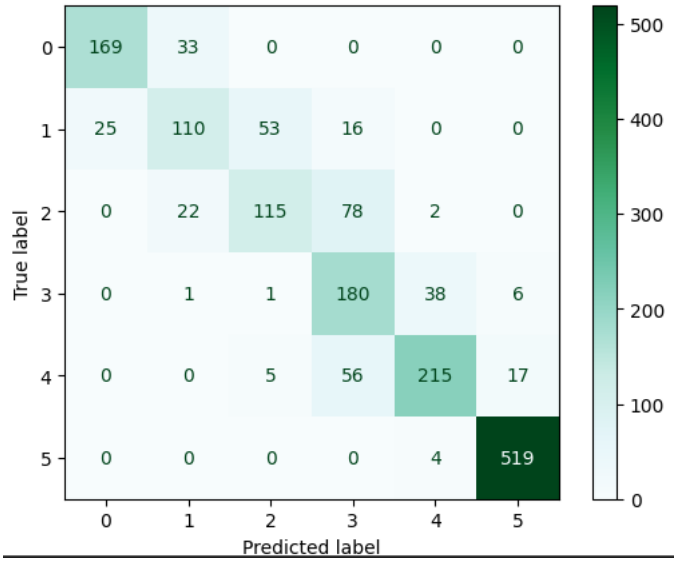

Figure 5: Confusion Matrix of the Decision Tree

2.7.1 Results

Analysing this confusion matrix we can see that the majority of the predicted are correct and even if they are not correct they aren't too far away from the correct result because it shows a lot of 0s arounds the confusion matrix, and that means if the model predicts that a person is Obese type I that result will be close to the truth, so that person could not have insufficient weight or normal weight, but something closer to the obese type I. To conclude this model turned out really good with a good accuracy and a good performance as well.

3 Conclusion

In conclusion, this report presented an analysis of a dataset using three popular machine learning techniques: k-means clustering, logistic regression, and decision tree. We found that the k-means algorithm was able to effectively cluster the data into distinct groups, providing insights into patterns and relationships within the dataset. The logistic regression model demonstrated good performance in predicting the outcome of a binary classification problem, with an overall accuracy of 90%. Lastly, the decision tree method effectively classify instances with an accuracy of 78%. Overall, this report demonstrated the power of machine learning in uncovering hidden patterns and insights within complex data. Each of the three techniques used in this analysis had their own strengths and weaknesses, and the choice of algorithm should be based on the specific problem and dataset at hand. This report also highlighted the importance of evaluating the performance of a model and understanding the limitations of the analysis. The most difficult challenges were the vizualization of the models and the good use of pyspark because we never worked with this library. Future work could include expanding the dataset or incorporating other machine learning techniques to improve the analysis. Additionally, this report could be used as a basis for further research or for developing a predictive model for a real-world application.

References

[1] 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.

[2] PySpark. (n.d.). PySpark: Python API for Spark.

[3] Pyspakr: AssembleVector

[4] Pyspark: Standardization

[5] Pyspark: Logistic Regression

[6] Pyspark: K-means Clustering

[7] Pyspark: Decision Tree

[8] Schikit-Learn: Confusion Matrix

Appendices

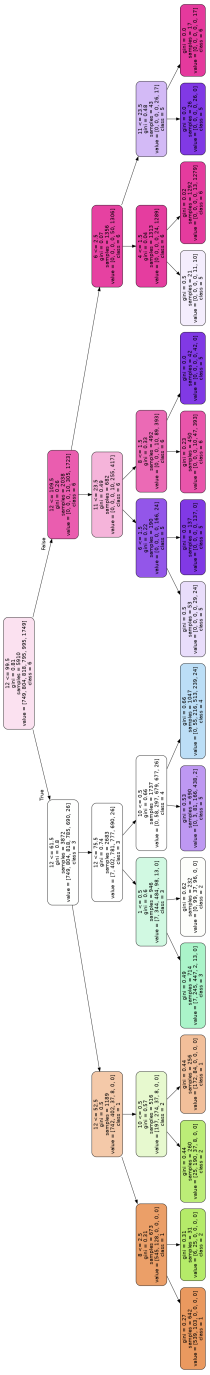

Figure 6: Decision Tree

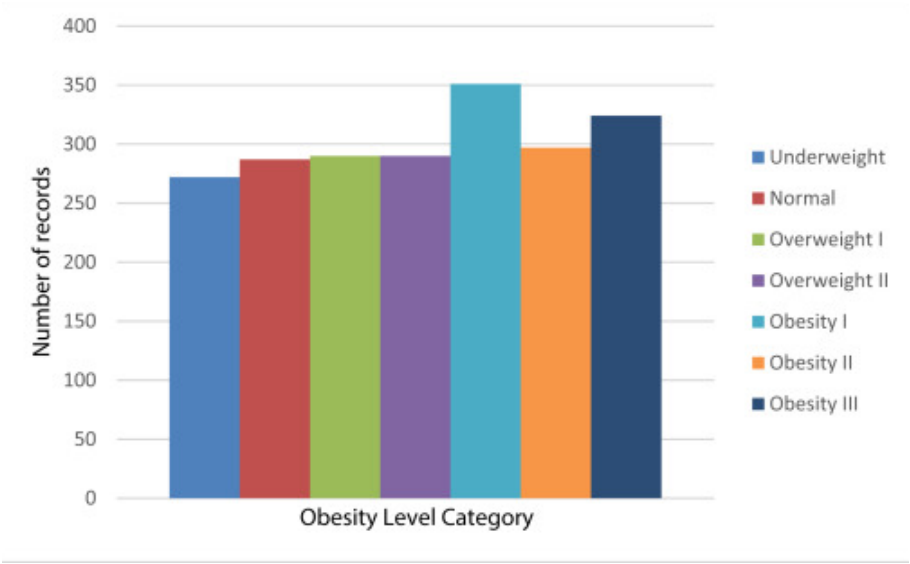

Figure 7: Balanced graphic

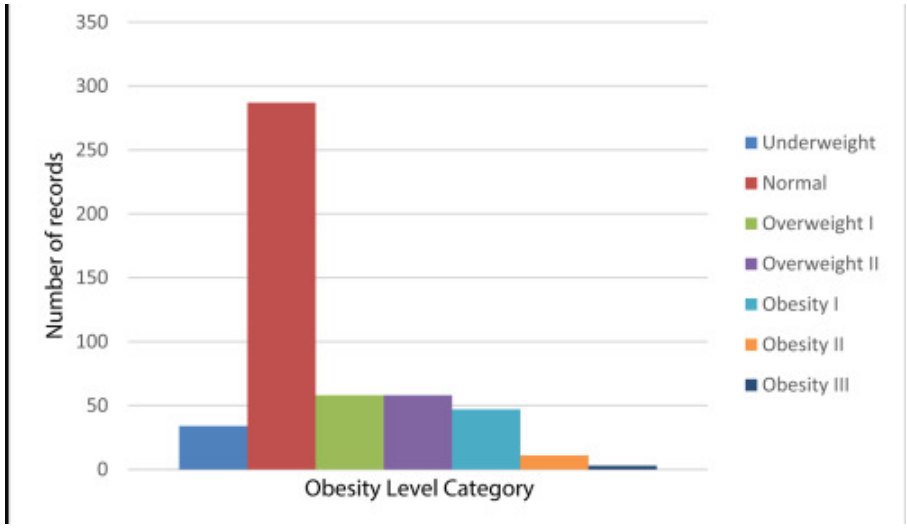

Figure 8: Non-balanced graphic