Python Project

By Gatien FOURNIER and Hippolyte GUESDON-VENNERIE

1) Data-Visualization

- First of all, here are the librairies that we used to study this dataset.
- We renamed the columns of the dataset so it will be easier to manipulate them.

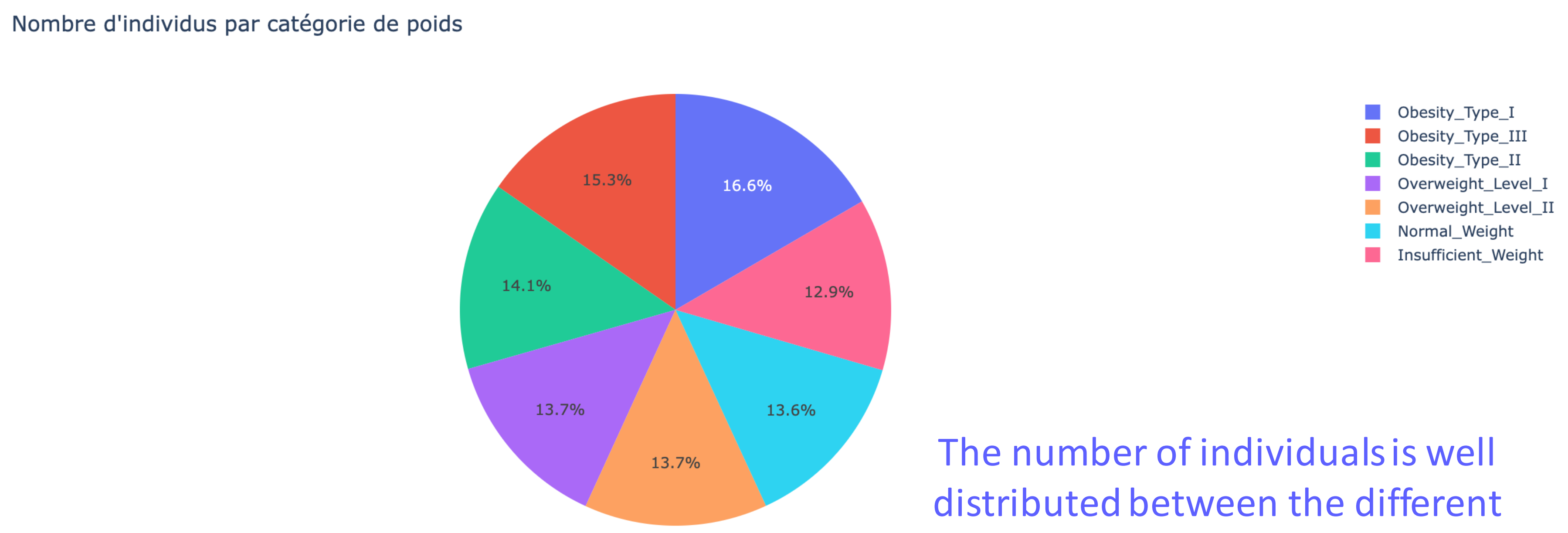
```
import seaborn as sns # used for plot interactive graph.
import numpy as np # linear algebra
import pandas as pd # data manipulation
import matplotlib.pyplot as plt # this is used for plot and graph
import seaborn as sns # used for plot interactive graph.
```

```
[199]
df=pd.read_csv("ObesityDataSet_raw_and_data_sinthetic.csv")
df.columns=["Gender", "Age", "Height", "Weight", "Family_history",
            "High_caloric_food", "Vegetables" , "number_meals", "Snacking",
            "Smoke", "Daily_water", "Count_calories", "Physical_activity",
            "Hours_screen", "Alcool", "Traveling_way", "Shape"]
```

First visualization of the dataset

Our dataset is an estimation of obesity levels based on different paremeters.

```
[105] fig = px.pie(df, names='Shape', title="Nombre d'individus par catégorie de poids")
      fig.show()
```

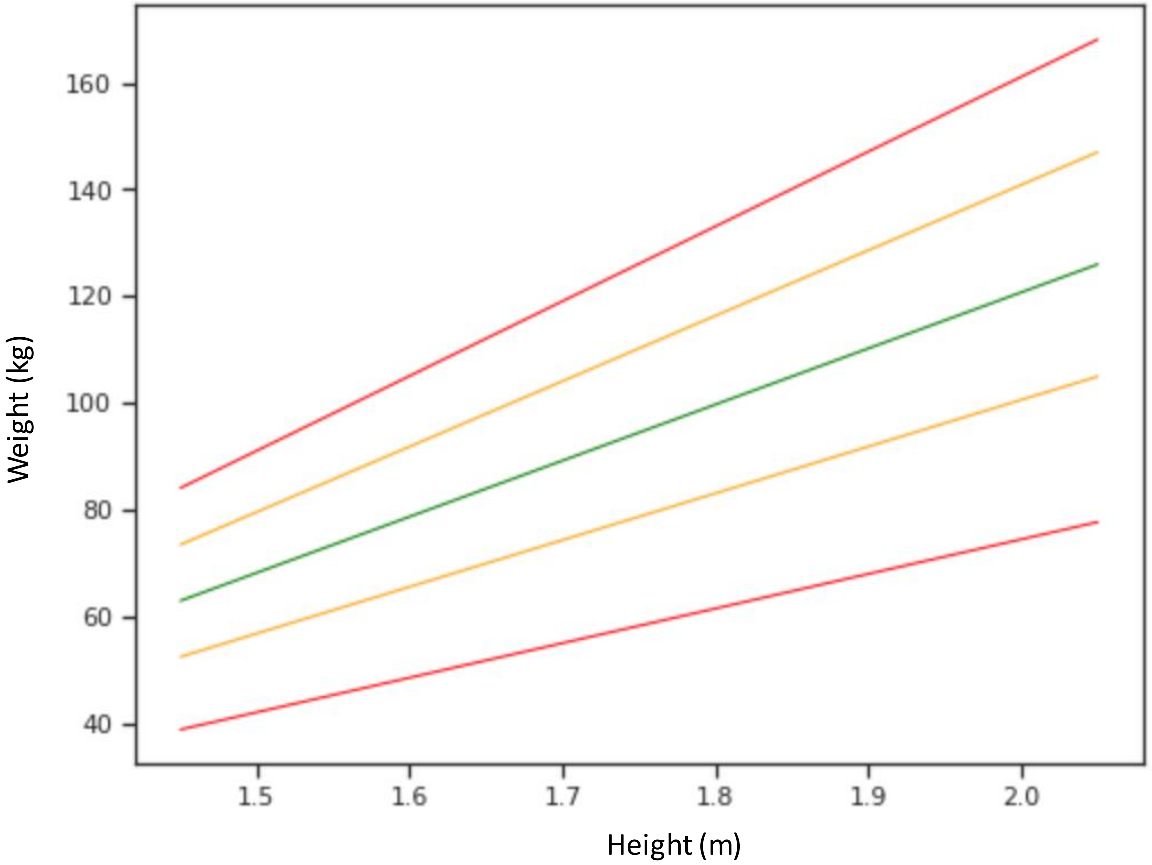


The number of individuals is well distributed between the different morphologies

Here we will try to visualize the BMI aspect on a graph with weight in function of height. This has nothing to see with the data from dataset, but it will help for our analysis

```
[229] def pointsimc(x, imc): #Fonction used to have the coordinates of the IMC limits
          y=imc*x*x
          return (x,y)
```

```
f, ax = plt.subplots(figsize=(8, 6))

xunderweight=[1.45,2.05] #Min and Max heights in the dataset
yunderweight=[38.89,77.746] #calculated coordinates for each cathegory of BMI
figure1= plt.plot(xunderweight, yunderweight, c="red", linewidth =1)

xnormal=[1.45,2.05] #Min and Max heights in the dataset
ynormal=[52.5,105] #calculated coordinates for each cathegory of BMI
figure1= plt.plot(xnormal, ynormal, c="orange", linewidth =1)

xoverweight=[1.45,2.05] #Min and Max heights in the dataset
yoverweight=[63,126] #calculated coordinates for each cathegory of BMI
figure1= plt.plot(xoverweight, yoverweight, c="green", linewidth =1)

xobesity2=[1.45,2.05] #Min and Max heights in the dataset
yobesity2=[73.5,147] #calculated coordinates for each cathegory of BMI
figure1= plt.plot(xobesity2, yobesity2, c="orange", linewidth =1)

xobesity3=[1.45,2.05] #Min and Max heights in the dataset
yobesity3=[84.1,168.1] #calculated coordinates for each cathegory of BMI
figure1= plt.plot(xobesity3, yobesity3, c="red", linewidth =1)
```

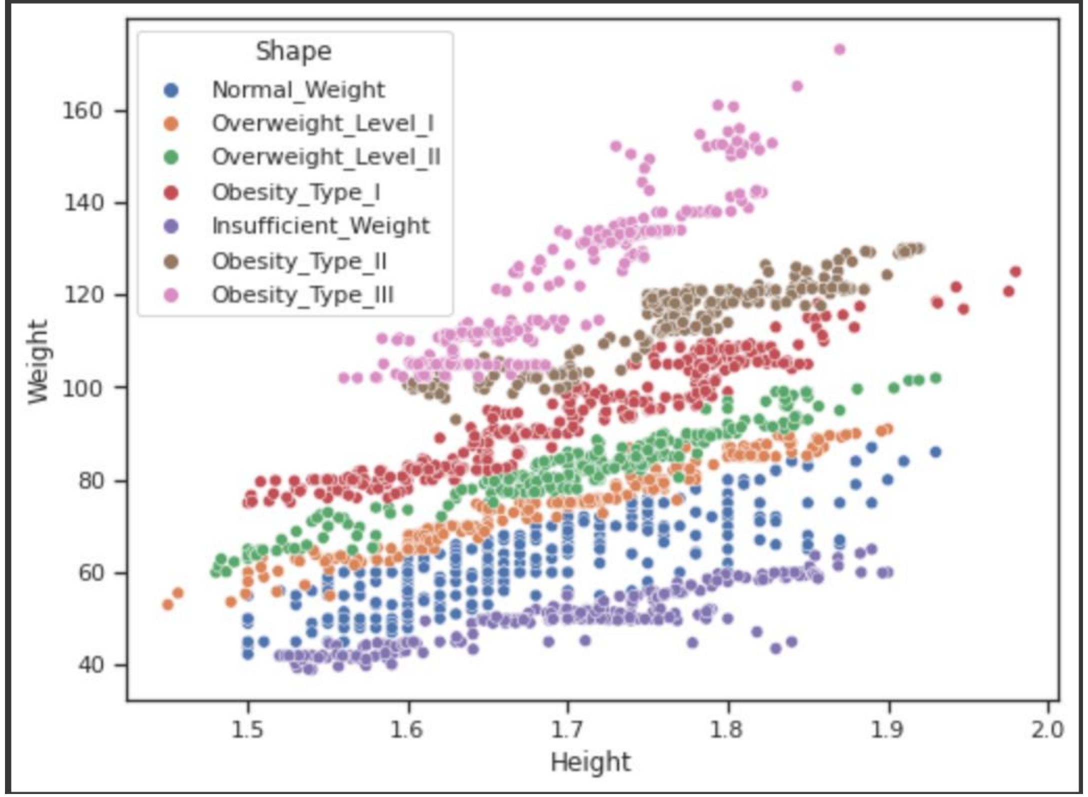


Let’s see if this is relevent

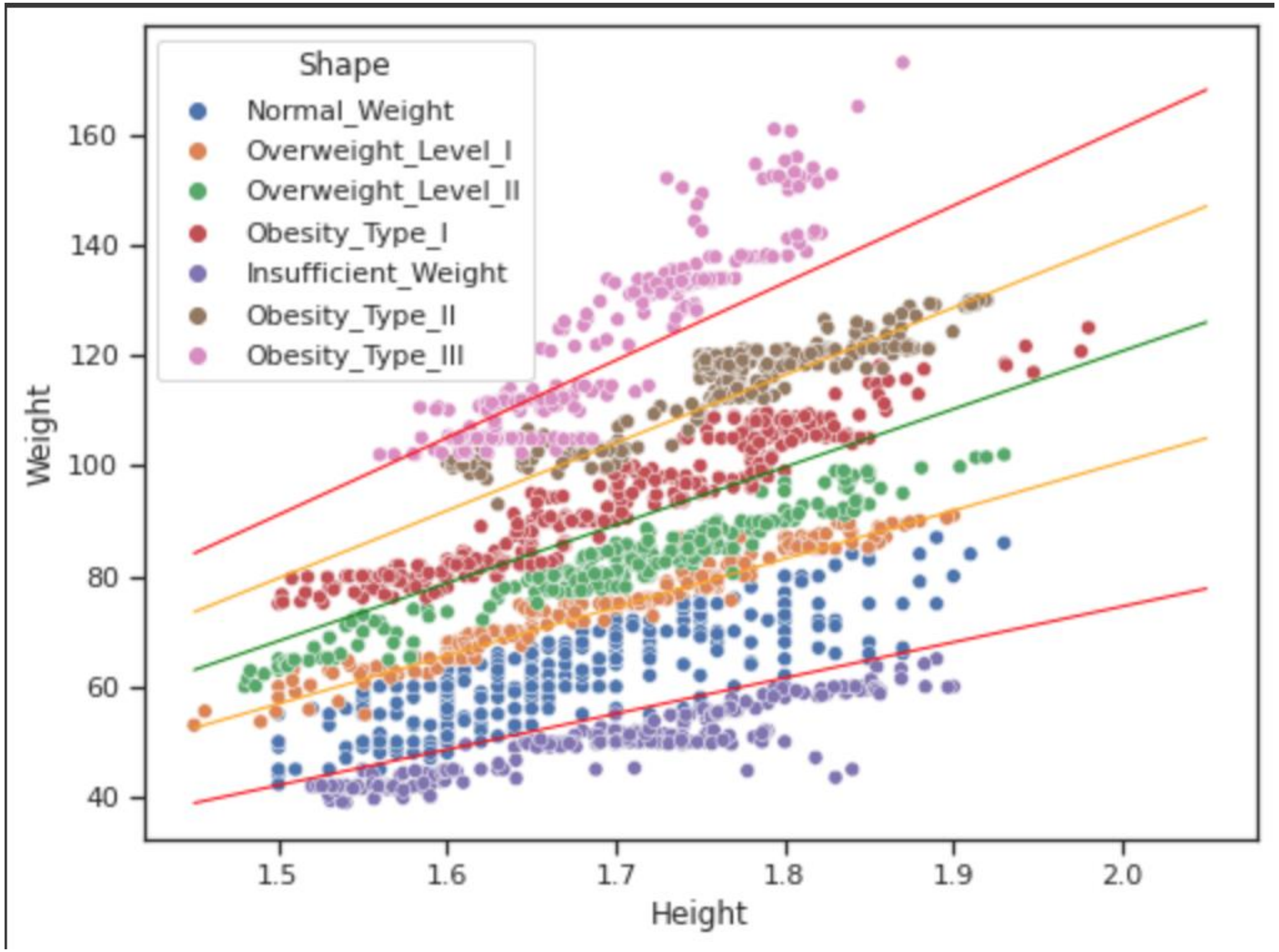
Comparing to the data, BMI sections go the same way as the obesity level classification !

Both graphs looks quite the same.

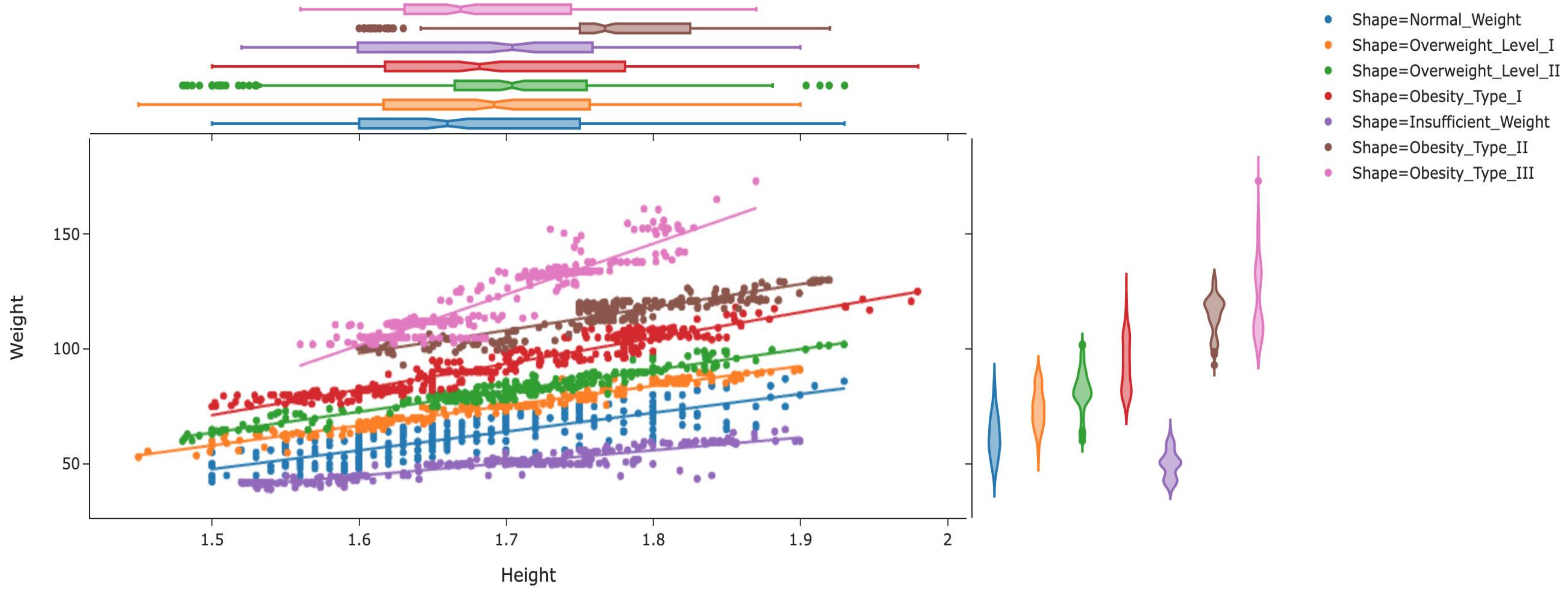
```
f, ax = plt.subplots(figsize=(8, 6))

sns.scatterplot(x='Height', y='Weight',
                hue='Shape', data=df);
```



Let’s combine them to see it clearer !



And here we see the real graph, with practical data, BMI lines are not right especially the pink one.

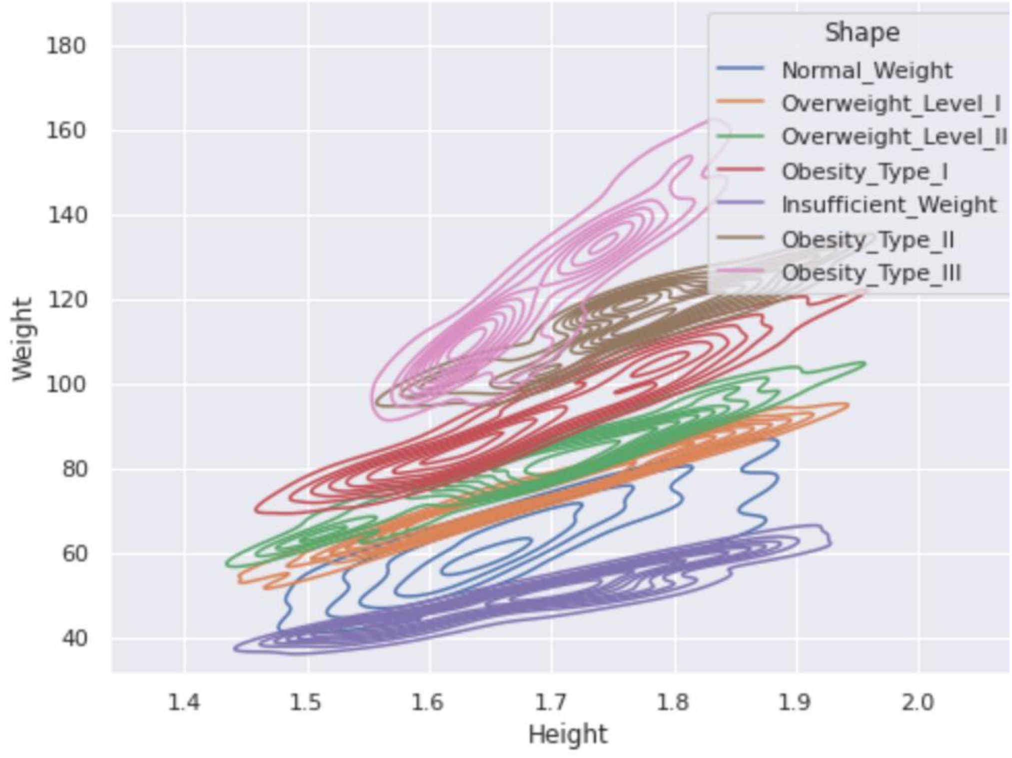


It seems that there is a little problem

- On this plot showing the density of each shape, we can clearly see that they do overlap.
- This is not supposed to happen because the result is determined only with the BMI using weight and height so the limits of each shape should be exact.
- However, the information about the dataset says that the shape is only determined with BMI (height and weight). There is a contradiction.

```
# Set up the figure
f, ax = plt.subplots(figsize=(8, 6))
# Draw a contour plot to represent each bivariate density
sns.kdeplot(
    data=df,
    x="Height",
    y="Weight",
    hue="Shape",)
```

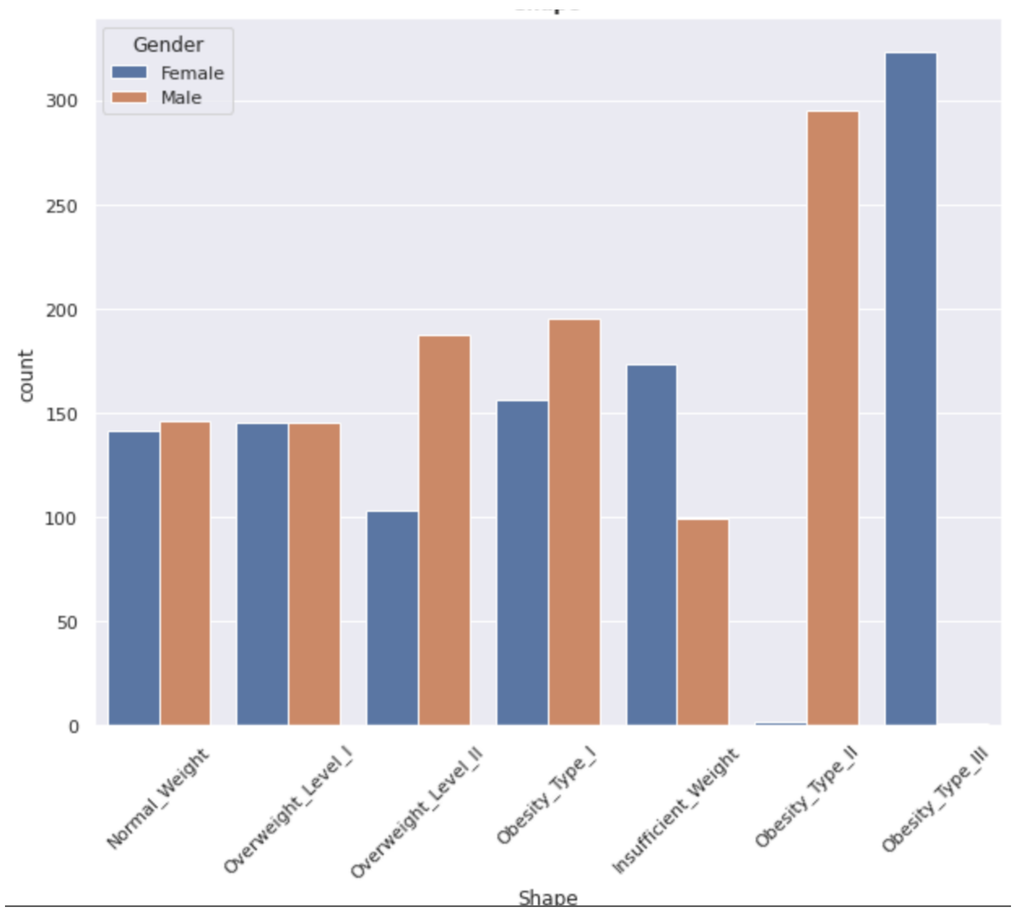
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fca602fea90>
```



Another anomaly in the dataset

It appears on this graph, we can see that almost only men are affected by obesity level 2 and on the contrary only women are affected by obesity level 3

```
f, ax = plt.subplots(figsize=(10, 8))

sns.countplot(x= 'Shape', hue = 'Gender',data = df)
plt.title('Shape', weight='bold')
plt.xticks(rotation=45)
```

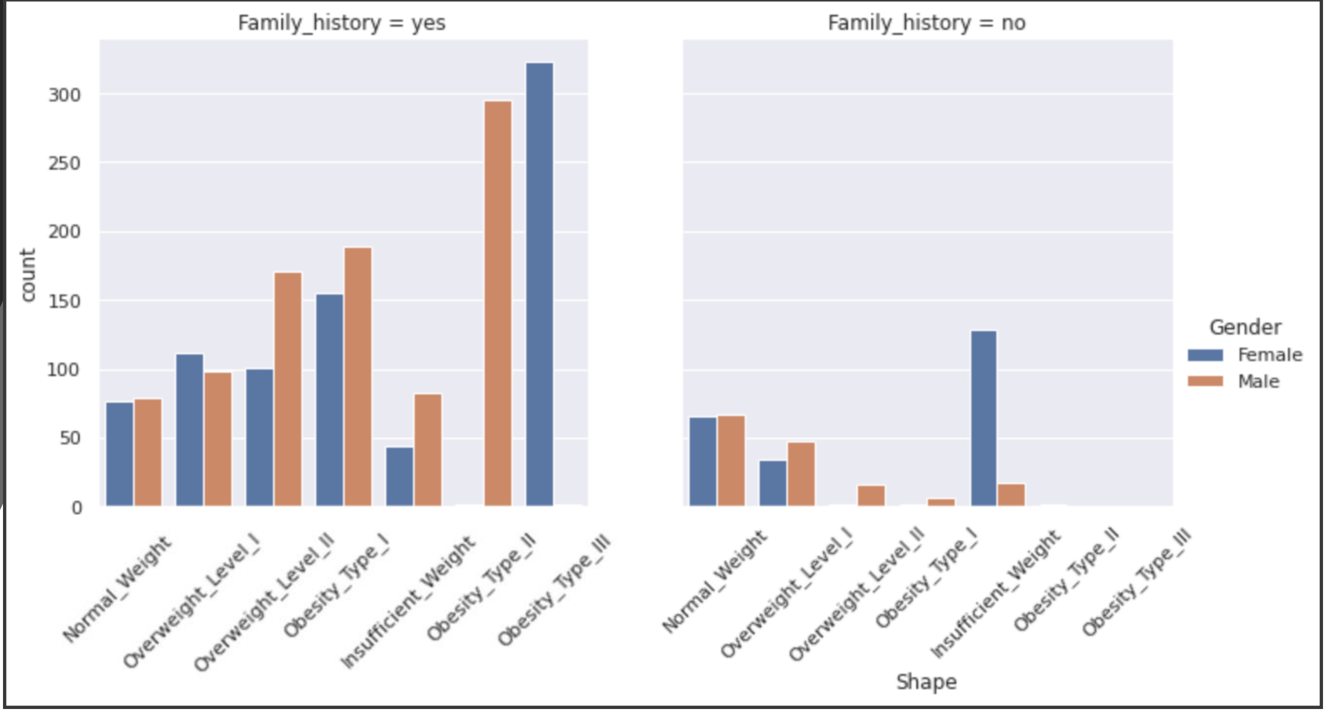


Family history, a major factor in obesity

- We quickly notice that the more obese you are, the more often you have a family history, so much so that only those with a family history were able to have level 2 or 3 obesity in this database.
- However, no family history seems to involves more insufficient weight

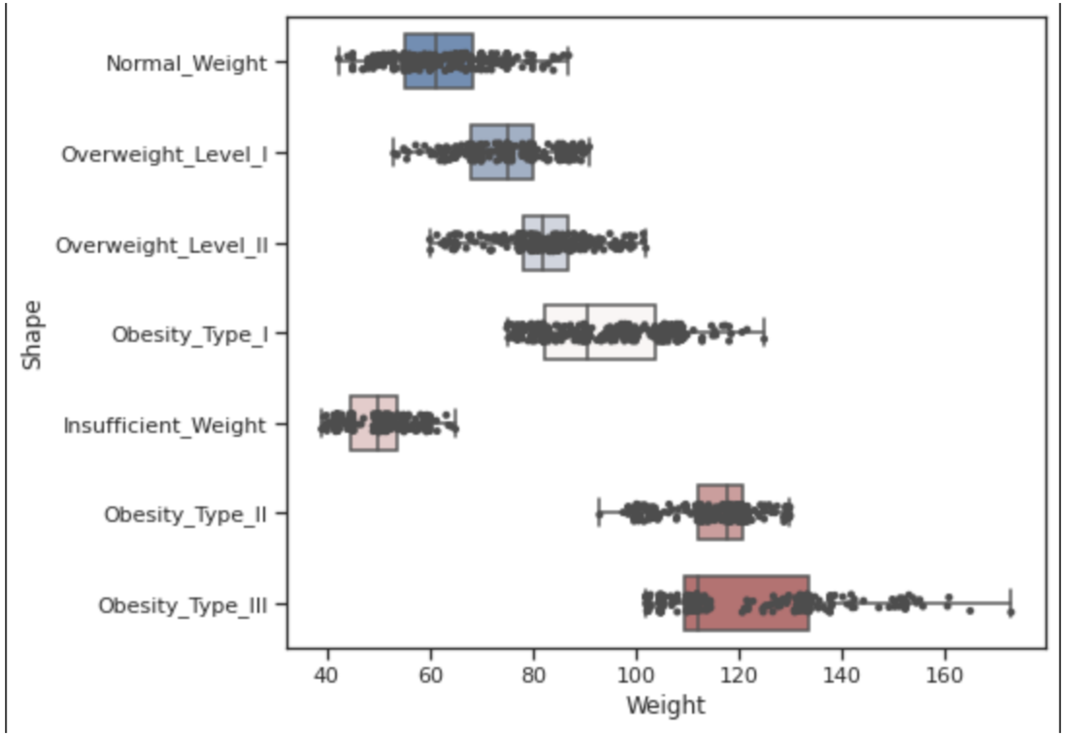
```
sns.catplot(x='Shape', hue="Gender", col="Family_history", data = df, kind="count");
plt.xticks(rotation=45)
```

```
(array([0, 1, 2, 3, 4, 5, 6]), <a list of 7 Text major ticklabel objects>)
```



BOXPLOTS

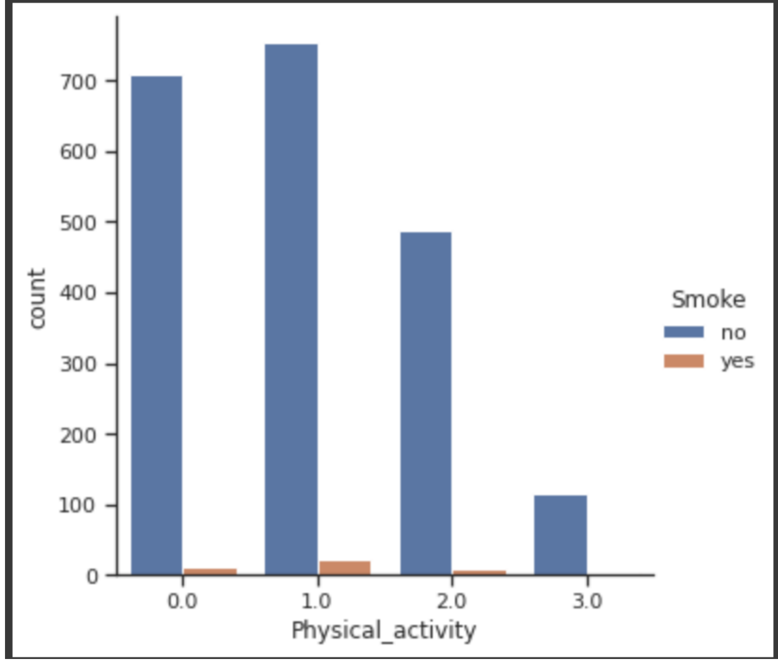
- Boxplots show us that if we get the weight of someone, we can already tell with a high efficiency in which categories can be the person.




```
sns.set_theme(style="ticks")
# Initialize the figure with a logarithmic x axis
f, ax = plt.subplots(figsize=(7, 6))
# Plot the orbital period with horizontal boxes
sns.boxplot(x="Weight", y="Shape", data=df,
            whis=[0, 100], width=.6, palette="vlag")
# Add in points to show each observation
sns.stripplot(x="Weight", y="Shape", data=df,
              size=4, color=".3", linewidth=0)
# Tweak the visual presentation
ax.xaxis.grid(True)
ax.set(ylabel="")
sns.despine(trim=True, left=True)
```

Smoking and physical activity

```
dfsport=df.round({'Physical_activity': 0})

sns.catplot(x='Physical_activity', hue="Smoke", data = dfsport, kind="count");
```

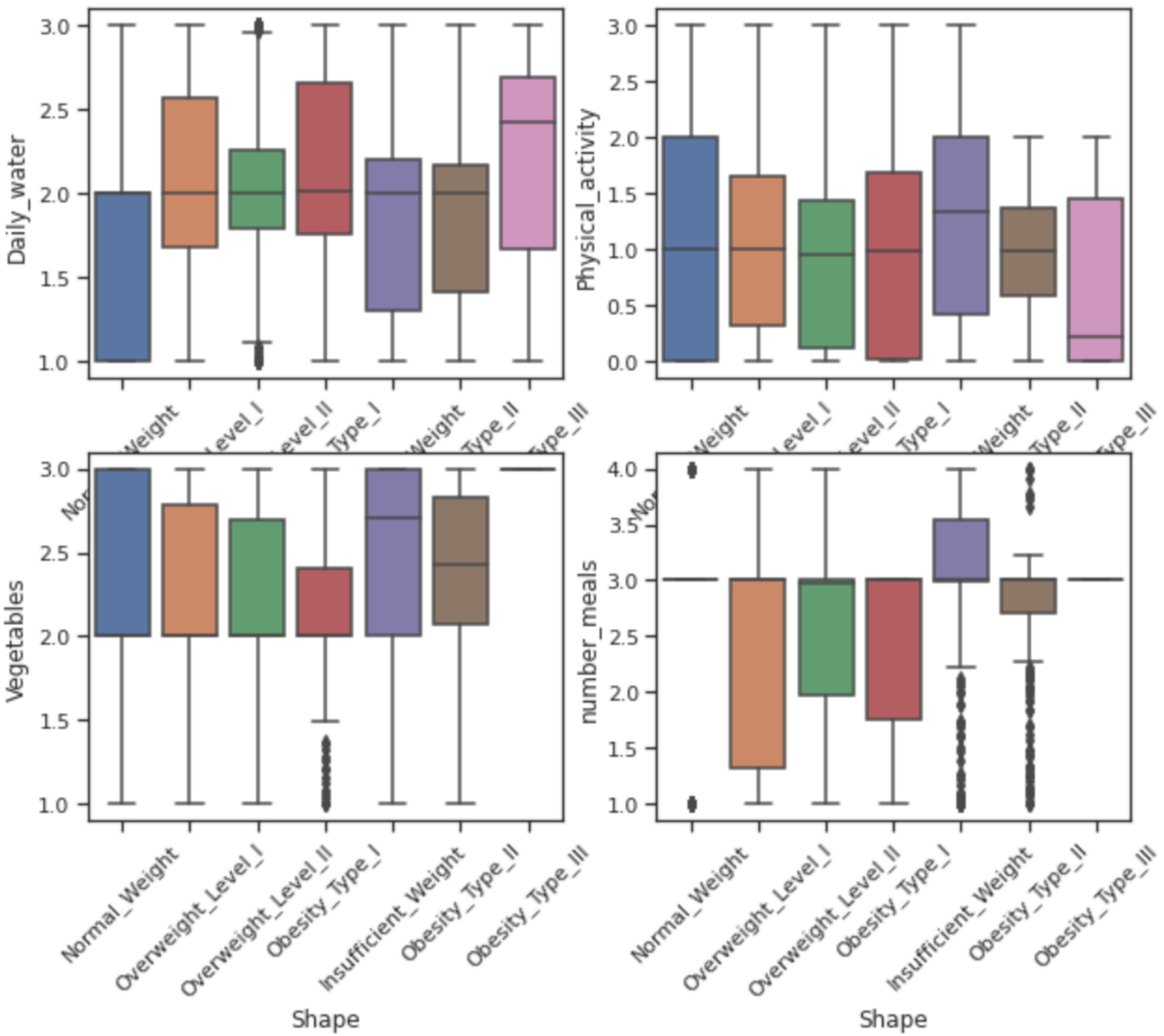


- - People doesn’t smoke a lot in this dataset, only 44 smokers out of nearly 2200 !
- - Also, we can see that no smoker play sports more than twice a week.

Comparison of a precise caracteristic between different obesity levels

We can see that they are all usefull in their way.

```
[115] f, ax = plt.subplots(figsize=(10,8))
      plt.subplot(221)
      sns.boxplot(x="Shape", y="Daily_water", data=df)
      plt.xticks(rotation = 45)
      plt.subplot(222)
      sns.boxplot(x="Shape", y="Physical_activity", data=df)
      plt.xticks(rotation = 45)
      plt.subplot(223)
      sns.boxplot(x="Shape", y="Vegetables", data=df)
      plt.xticks(rotation = 45)
      plt.subplot(224)
      sns.boxplot(x="Shape", y="number_meals", data=df)
      plt.xticks(rotation = 45)
```



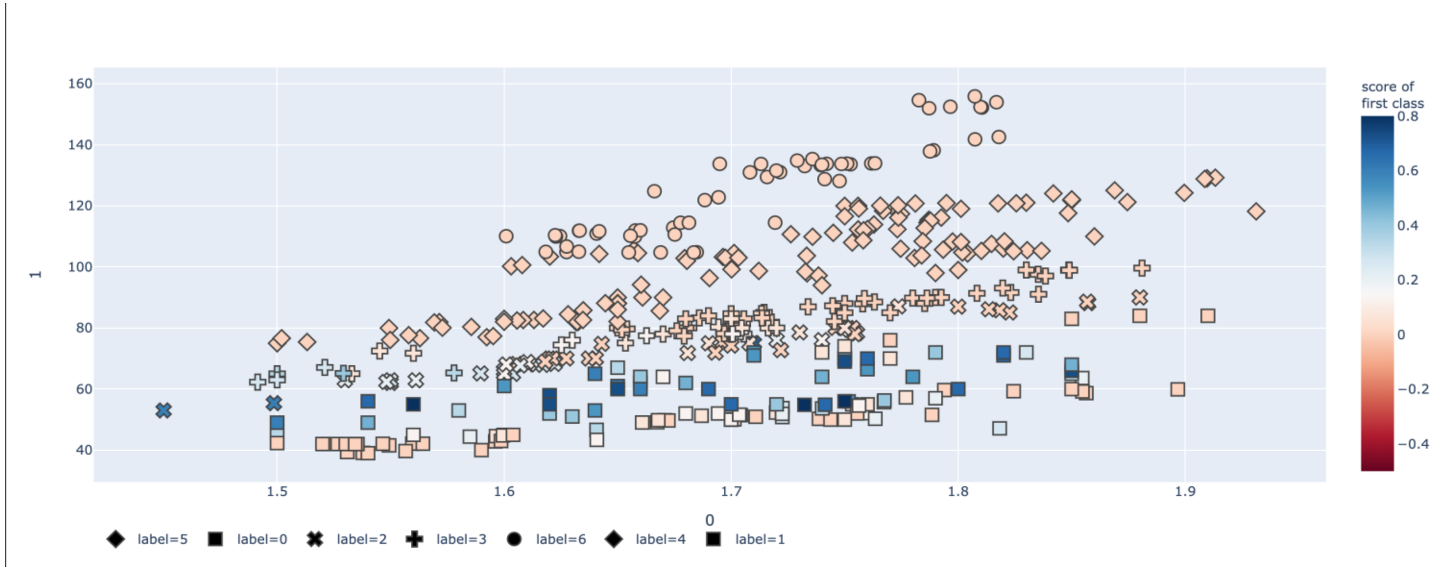
2) Data classification

- In this part, we will use different machine learning process.
- First of all, we have to encode to just have numeric values.

K- Neighbors

```
import plotly.express as px
import numpy as np
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

data=data_dummies.drop(['Age'],axis=1)

X = data.iloc[:, data.columns != 'Shape'].values
y = data.iloc[:, 7].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

# Fit the model on training data, predict on test data
clf = KNeighborsClassifier(15)
clf.fit(X_train, y_train)
y_score = clf.predict_proba(X_test)[:, 1]

fig = px.scatter(
    X_test, x=0, y=1,
    color=y_score, color_continuous_scale='RdBu',
    symbol=y_test, symbol_map={'0': 'square-dot', '1': 'circle-dot', '2': 'circle',
                               '3': 'square', '4': 'cross-dot', '5': 'cross'},
    labels={'symbol': 'label', 'color': 'score of <br>first class'}
)
fig.update_traces(marker_size=12, marker_line_width=1.5)
fig.update_layout(legend_orientation='h')
fig.show()
```

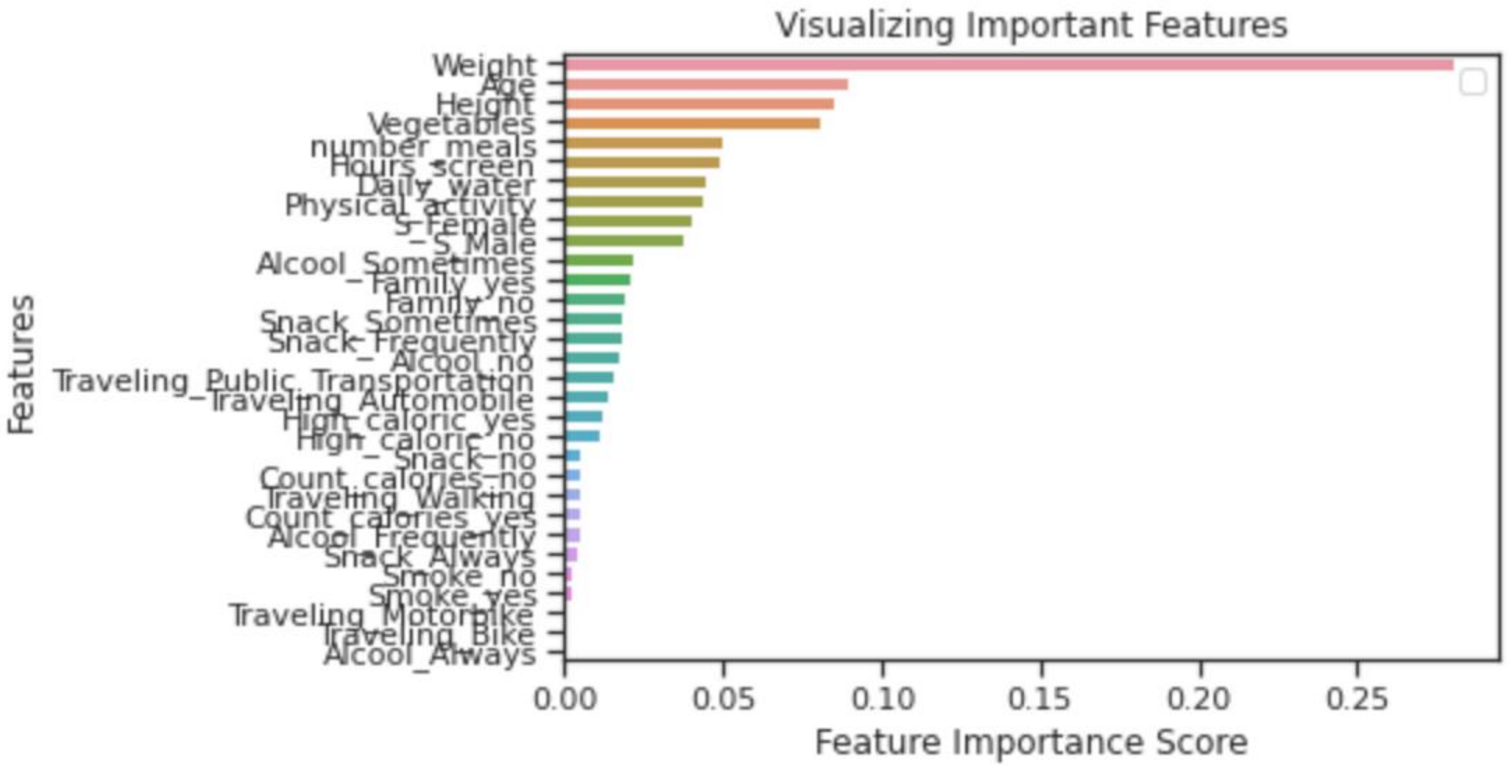
K- Neighbors




```
[[51  5  1  0  2  0  0]
 [ 8 25  8  7  2  2  0]
 [ 1  8 41  4  1  0  0]
 [ 1  5  5 42  2  0  1]
 [ 1  3  1  2 55  6  1]
 [ 0  1  0  0  0 58  0]
 [ 0  1  1  0  0  0 71]]
              precision    recall  f1-score   support

           0       0.82      0.86      0.84        59
           1       0.52      0.48      0.50        52
           2       0.72      0.75      0.73        55
           3       0.76      0.75      0.76        56
           4       0.89      0.80      0.84        69
           5       0.88      0.98      0.93        59
           6       0.97      0.97      0.97        73

    accuracy                           0.81       423
   macro avg       0.79      0.80      0.80       423
weighted avg       0.81      0.81      0.81       423
```

Here is the visualized results of the classification. It actually worked well the accuracy is good and the representation looks like the one we had at the start.

Random Forest




```
[161] f, ax = plt.subplots(figsize=(10,8))
      import matplotlib.pyplot as plt
      import seaborn as sns
      %matplotlib inline
      # Creating a bar plot
      sns.barplot(x=feature_imp, y=feature_imp.index)
      # Add labels to your graph
      plt.xlabel('Feature Importance Score')
      plt.ylabel('Features')
      plt.title("Visualizing Important Features")
      plt.legend()
      plt.show()
```

We can say with no surprise that the weight is the most important factor of obesity, age height and eating vegetable are also very important.

```
22/22 [==============================] - 0s 1ms/step - loss: -523756.2188 - accuracy: 0.1478
[-523756.21875, 0.14777618646621704]
```

```
Epoch 1/8
1414/1414 [==============================] - 3s 2ms/step - loss: -485.0847 - accuracy: 0.1266
Epoch 2/8
1414/1414 [==============================] - 2s 2ms/step - loss: -7638.5488 - accuracy: 0.1301
Epoch 3/8
1414/1414 [==============================] - 2s 2ms/step - loss: -28942.5684 - accuracy: 0.1301
Epoch 4/8
1414/1414 [==============================] - 2s 2ms/step - loss: -67270.6953 - accuracy: 0.1301
Epoch 5/8
1414/1414 [==============================] - 2s 2ms/step - loss: -125427.8672 - accuracy: 0.1301
Epoch 6/8
1414/1414 [==============================] - 2s 2ms/step - loss: -206218.9531 - accuracy: 0.1301
Epoch 7/8
1414/1414 [==============================] - 2s 2ms/step - loss: -310853.0625 - accuracy: 0.1301
Epoch 8/8
1414/1414 [==============================] - 2s 2ms/step - loss: -443123.0938 - accuracy: 0.1301
<keras.callbacks.History at 0x7f52c86152d0>
```

Neural Network

The neural network was very hard to set up. The results obtained are very bad. We see an accuracy of 14% which means that the code does not work. We would have liked with more time to understand the reason for this failure.

Thank you for your attention