Estimation of obesity levels based on eating habits and physical condition Data Set

Thomas CULINO, Zachary CHENOT, DIA3

Data Visualization

Data importation

- We imported the dataset without any problems. There weren't any missing values that we had to replace.

```
df = pd.read_csv("ObesityDataSet_raw_and_data_sinthetic.csv")
df.head()
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	M
0	Female	21.0	1.62	64.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	0.0	1.0	no	Public_Transp
1	Female	21.0	1.52	56.0	yes	no	3.0	3.0	Sometimes	yes	3.0	yes	3.0	0.0	Sometimes	Public_Transp
2	Male	23.0	1.80	77.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	2.0	1.0	Frequently	Public_Transp
3	Male	27.0	1.80	87.0	no	no	3.0	3.0	Sometimes	no	2.0	no	2.0	0.0	Frequently	
4	Male	22.0	1.78	89.8	no	no	2.0	1.0	Sometimes	no	2.0	no	0.0	0.0	Sometimes	Public_Transp

Data Processing

- The first thing we did after that was explaining what each variable was doing. It was hard renaming them because it was difficult to reduce their function to one or two words.

Explaination of each variable

Gender : gender of the person
Age : age of the person
Height : height of the person
Weight : weight of the person
family_history_with_overweight : if the person has someone in his family who was in overweight
FAVC : if the person eats caloric food frequently
FCVC : if the person eats vegetables in his meals
NCP : number of meals a day
CAEC : if the person eats between meals
SMOKE : if the person smokes
CH2O : the quantity of water the person drinks daily
SCC : if the person monitors the calories they eat daily
FAF : frequency of physical activity in a week
TUE : Time spent daily on technological devices
CALC : frequency of drinking alcohol
MTRANS : Means of transport majoritarly used by the person
NObeyesdad : target variables which corresponds to the BMI (Body Mass Index)

Data processing

- We converted all ‘string’ type variables to ‘category’ type.
- We did it to use “pandas.get_dummies” later on the machine learning part later
- Moreover, this save some memory so it was beneficial for us.

```
#We convert non-quantitative variables to type "category"
columns = ["Gender", "family_history_with_overweight", "FAVC", "FCVC", "CAEC", "SMOKE", "SCC", "CALC", "MTRANS", "NObeye

for col in columns:
    df[col] = df[col].astype('category')
```

- We also mapped the ‘FCVC’ variable, which corresponds to how frequently the person eat vegetables in his meals? to ‘Never’, ‘Sometimes’ and ‘Always’ as it was the rightful values.

```
df['FCVC'] = df['FCVC'].map({1:'Never', 2:'Sometimes', 3:'Always'})
```

Data processing

- We then chose to change floats to integers because we wanted the data to be clear. In some cases, we had float variables (ex: CH2O) which corresponded to integer values. We modified this to have simpler and more readable data.
- But that also suits us in another way because of the performance and efficiency. Integer arithmetic is faster. And for a given range integers consume less memory because integers don't need to represent non-integer values.

```
#On convertit les variables float au int le plus proche
columns = ["NCP", "CH2O", "TUE", "FAF"]

for col in columns:
    df[col] = round(df[col]).astype('int')

df.head()
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	
0	Female	21.0	1.62	64.0	yes	no	Sometimes	3	Sometimes	no	2	no	0	1	no	Public_T
1	Female	21.0	1.52	56.0	yes	no	Always	3	Sometimes	yes	3	yes	3	0	Sometimes	Public_T
2	Male	23.0	1.80	77.0	yes	no	Sometimes	3	Sometimes	no	2	no	2	1	Frequently	Public_T
3	Male	27.0	1.80	87.0	no	no	Always	3	Sometimes	no	2	no	2	0	Frequently	
4	Male	22.0	1.78	89.8	no	no	Sometimes	1	Sometimes	no	2	no	0	0	Sometimes	Public_T

Data processing

- After that, we looked at some more information regarding the dataset such as its description and its information.

```
1 df.shape
```

```
(2111, 17)
```

```
1 df.describe()
```

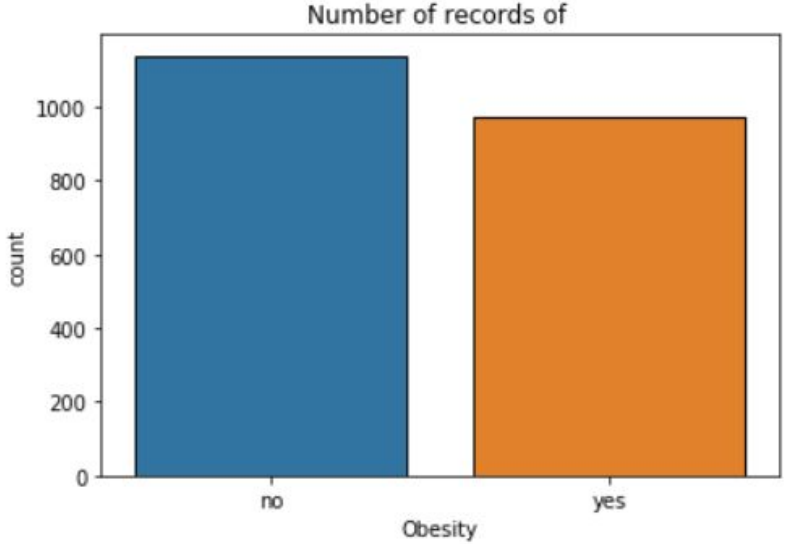
	Age	Height	Weight	NCP	CH2O	FAF	TUE
count	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000
mean	24.312600	1.701677	86.586058	2.687826	2.014685	1.006632	0.664614
std	6.345968	0.093305	26.191172	0.809680	0.688616	0.895462	0.674009
min	14.000000	1.450000	39.000000	1.000000	1.000000	0.000000	0.000000
25%	19.947192	1.630000	65.473343	3.000000	2.000000	0.000000	0.000000
50%	22.777890	1.700499	83.000000	3.000000	2.000000	1.000000	1.000000
75%	26.000000	1.768464	107.430682	3.000000	2.000000	2.000000	1.000000
max	61.000000	1.980000	173.000000	4.000000	3.000000	3.000000	2.000000

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
Gender                            2111 non-null category
Age                               2111 non-null float64
Height                            2111 non-null float64
Weight                            2111 non-null float64
family_history_with_overweight    2111 non-null category
FAVC                              2111 non-null category
FCVC                              1285 non-null category
NCP                               2111 non-null int32
CAEC                              2111 non-null category
SMOKE                             2111 non-null category
CH2O                              2111 non-null int32
SCC                               2111 non-null category
FAF                               2111 non-null int32
TUE                               2111 non-null int32
CALC                              2111 non-null category
MTRANS                            2111 non-null category
NObeyesdad                        2111 non-null category
dtypes: category(10), float64(3), int32(4)
memory usage: 104.7 KB
```

Plotting

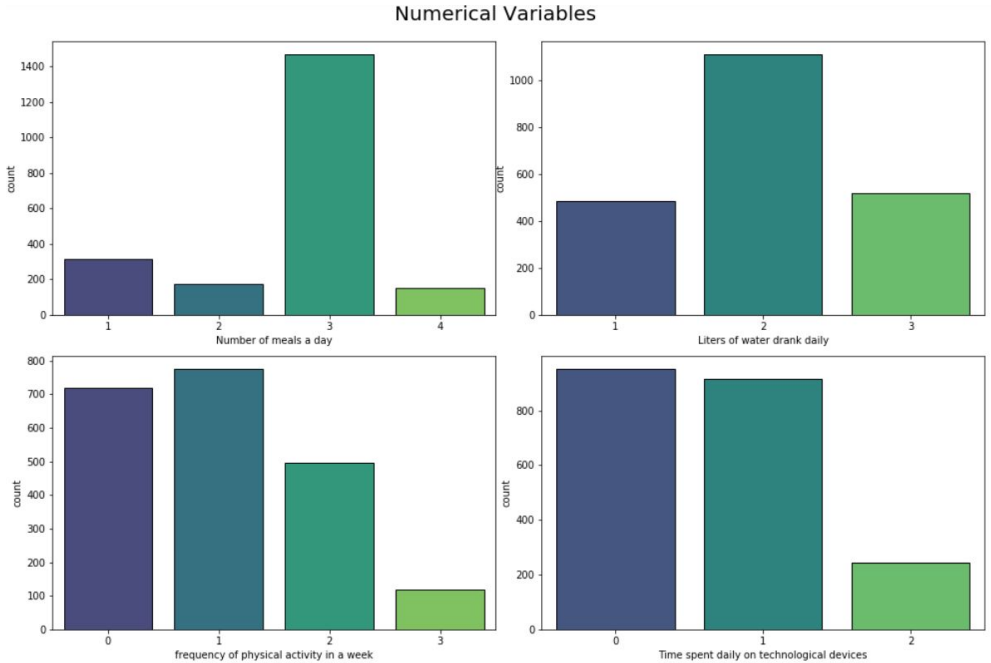
- We began plotting the data to see the count of people affected by obesity:



- As we can see, the dataset is quite homogeneous in the sense that the number of people affected by obesity is quite well distributed.

Plotting

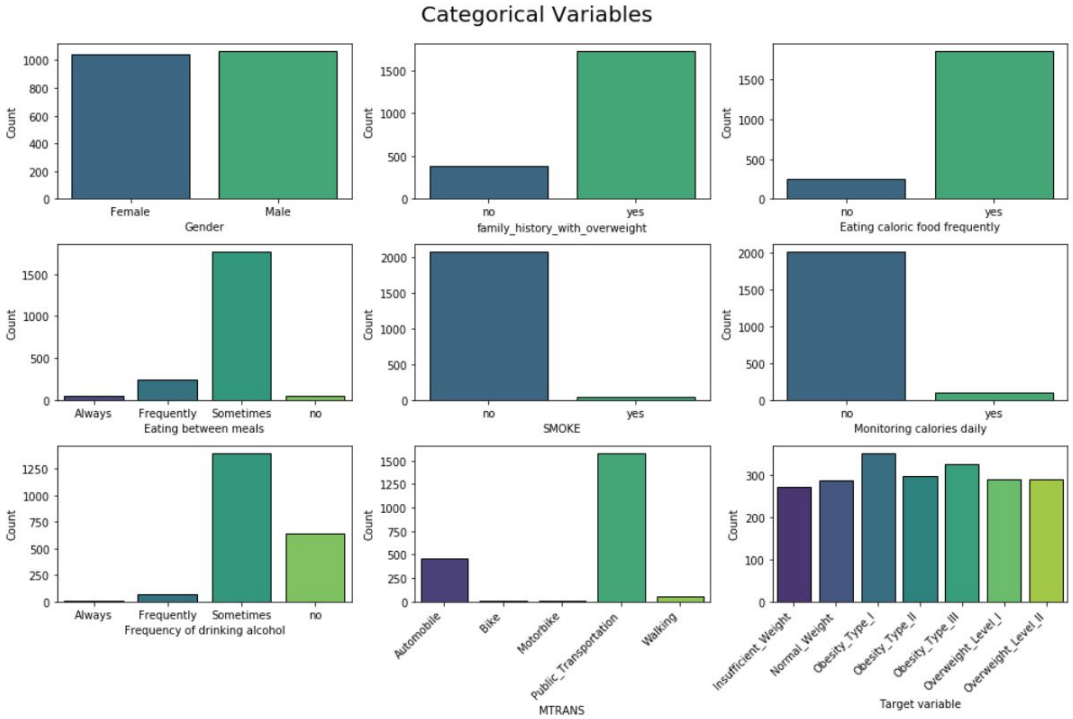
- Then, we wanted to see how numerical variables were distributed:



- We can see on the top left plot that the majority of people eat 3 meals a day. If we recall the previous plot, the number of people affected by obesity was almost evenly distributed. We can therefore assume that this variable is not preponderant in the prediction of our target variable.
- As for the bottom right plot, most people don't spend too much time on technological devices everyday. We can then assume that surexposition to technological devices is not the principal cause of overweight or obesity.

Plotting

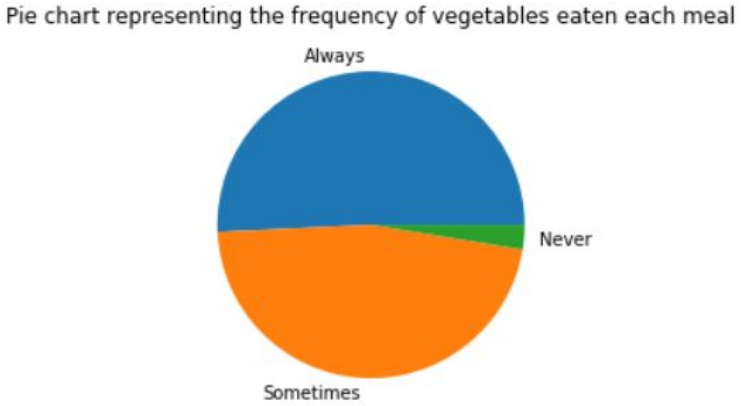
- After the numerical values, we wanted to see the trends of the categorical variables:



- At the bottom right corner, we can see the countplot of the target variable which is pretty much evenly distributed. As such, we can suppose that any disparity in the other countplots can affect our future model.
- We can see those disparities in all the plots from the 2nd to the 8th.

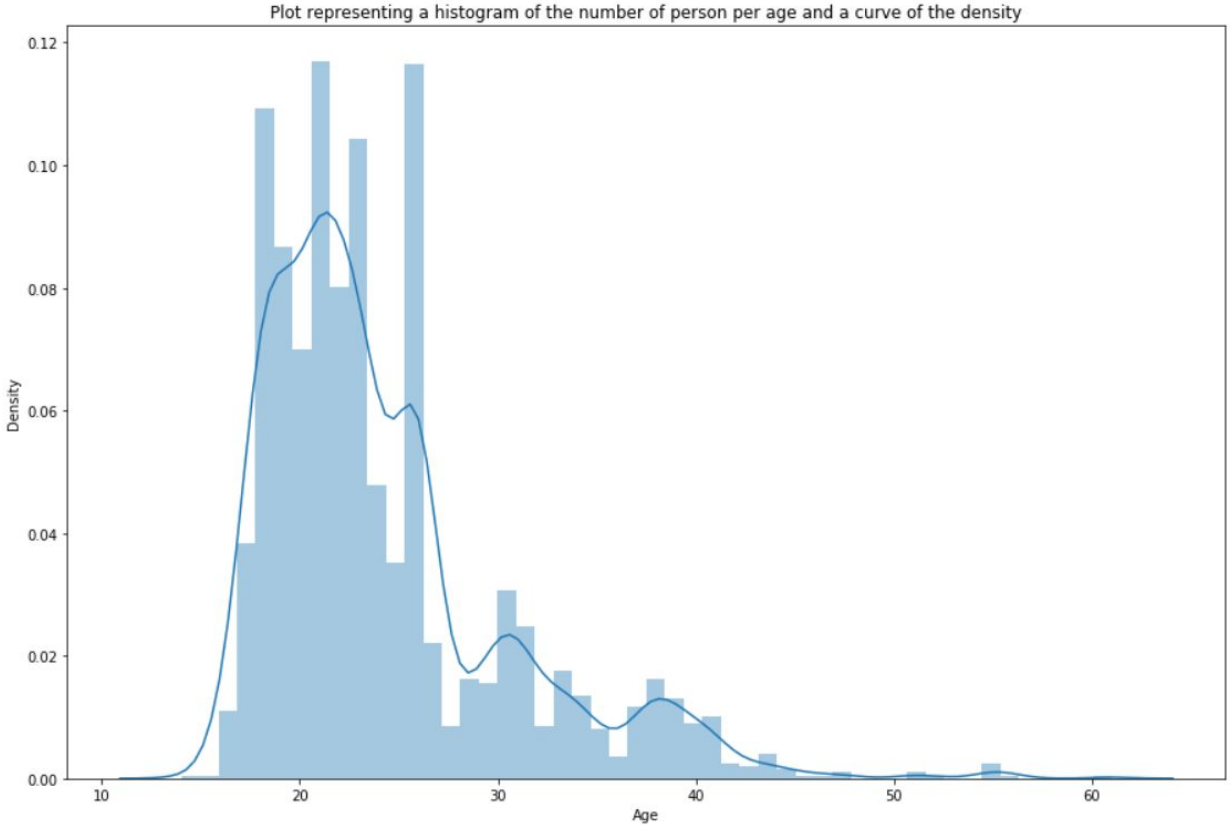
Plotting

- Because we had plotted all ‘category’ variables except ‘FCVC’ which is the frequency of vegetables eaten each meal, we wanted to make up for it.



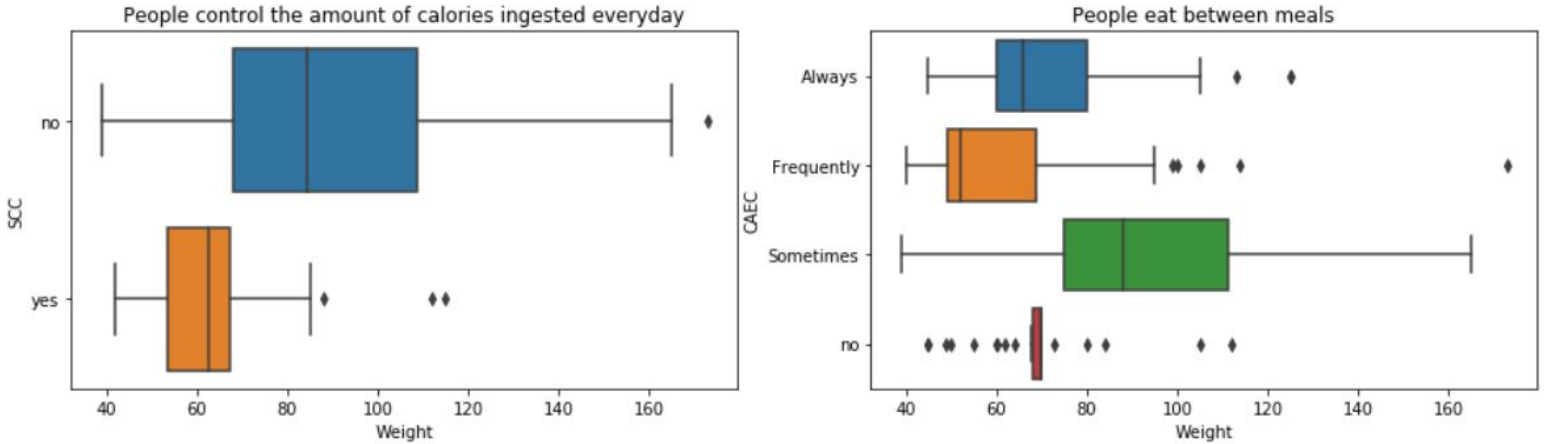
- As we can see on the pie chart, people never eating vegetables in their meals are a minority. We can thus assume that this variable does not account to much in the face of predicting weight categories. We can even assume that ‘FCVC’ and the target variable are not strongly related.

Plotting



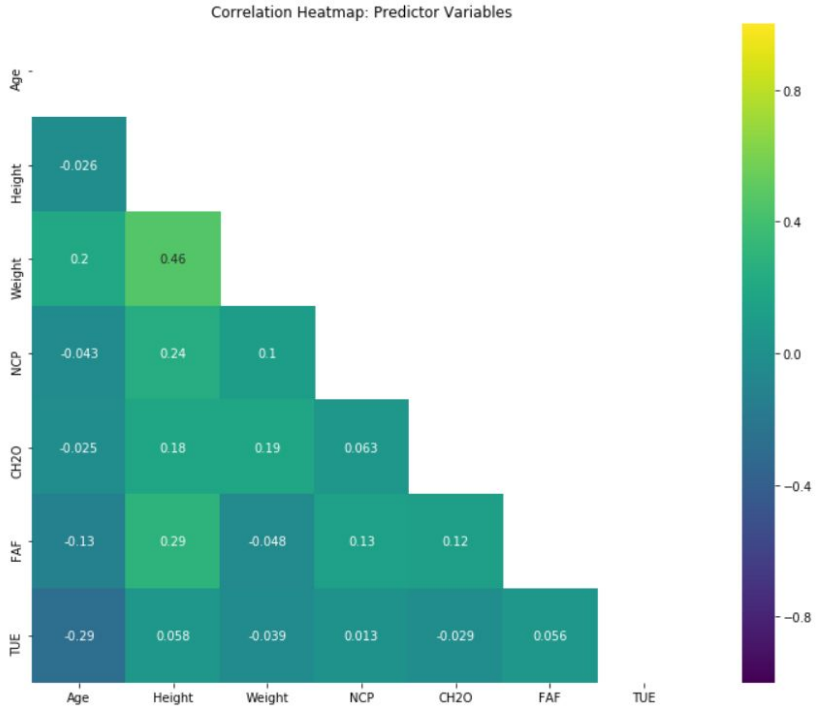
- This plot shows us that the people in the dataset are quite young, having the highest density of people around 22 years old.

Plotting



- On the left plot, we can see that the dispersion is bigger for people not controlling the amount of calories they ingest everyday. This shows that people controlling the amount of calories they ingest all have a weight concentrated around 60kg.
- On the right plot, we can definitely see a correlation between eating between meals and the weight. This is shown by the boxplot at the bottom displaying the weight of people who don’t eat between meals. Like the previous plot, their weight is concentrated around 70kg. However, even if we can see a correlation between a low weight and not eating between meals, the opposite is not true. In fact, there is not much difference between people always eating between meals and people doing it frequently.

Plotting



- On the heatmap, we can see which variables are more important as predictors. We can see that the Height is the most important variable followed by the Age. We are looking at the absolute value.

Machine Learning

Choice of variables and models

- First, we created a copy of our dataframe

```
1 df_prep = df.copy()
2 df_prep.head()
```

- After that, we tried running different models such as Random Forest, Decision Tree, KNN, SVM, and then we tuned KNN, Random Forest and Decision Tree.

- We thought for a while about whether we should save only the “Height” and “Weight” variables or not since it’s basically what we use to get the Body Mass Index (BMI).

Choice of variables and models

- After a conversation with you we ended up saying that the machine doesn't know anything about the BMI so it would still be considered as machine Learning.

- Then with the models chosen, we tried different combinations of variables to see which combination gave the best results:

```
All variables excepts Height and Weight:
knn:0.76498
random forest:0.78707
decision tree:76.341
svm:72.397

All the variables:
knn:88.17
random forest:91.167
decision tree:94.795
svm:56.94

All variables but creating age, weight and height group :
knn:87.224
random forest:89.117
decision tree:89.748
svm:78.707
```

- We concluded that the best results were found when we only used Height and Weight. We can see the results on the next slide

Results from our models of two variables (Height & Weight)

Random Forest:

Accuracy: 0.96057
Accuracy with Scaled Data: 0.95899

Classification Report:

	precision	recall	f1-score	support
Insufficient_Weight	0.97	0.98	0.97	92
Normal_Weight	0.93	0.92	0.93	77
Obesity_Type_I	0.95	0.98	0.97	114
Obesity_Type_II	0.97	0.98	0.97	85
Obesity_Type_III	1.00	0.99	0.99	92
Overweight_Level_I	0.94	0.93	0.94	89
Overweight_Level_II	0.96	0.93	0.95	85
accuracy			0.96	634
macro avg	0.96	0.96	0.96	634
weighted avg	0.96	0.96	0.96	634

KNN:

Accuracy: 0.79811
Accuracy with Scaled Data: 0.94795

Classification Report:

	precision	recall	f1-score	support
Insufficient_Weight	0.98	0.96	0.97	92
Normal_Weight	0.77	0.88	0.82	77
Obesity_Type_I	0.76	0.74	0.75	114
Obesity_Type_II	0.75	0.84	0.79	85
Obesity_Type_III	0.90	0.85	0.87	92
Overweight_Level_I	0.74	0.75	0.74	89
Overweight_Level_II	0.69	0.59	0.64	85
accuracy			0.80	634
macro avg	0.80	0.80	0.80	634
weighted avg	0.80	0.80	0.80	634

Decision Tree:

Accuracy: 0.93691
Accuracy with Scaled Data: 0.93533

Classification Report:

	precision	recall	f1-score	support
Insufficient_Weight	0.98	0.99	0.98	92
Normal_Weight	0.91	0.87	0.89	77
Obesity_Type_I	0.96	0.96	0.96	114
Obesity_Type_II	0.95	0.93	0.94	85
Obesity_Type_III	0.97	0.98	0.97	92
Overweight_Level_I	0.87	0.89	0.88	89
Overweight_Level_II	0.92	0.93	0.92	85
accuracy			0.94	634
macro avg	0.94	0.93	0.93	634
weighted avg	0.94	0.94	0.94	634

SVM:

Accuracy: 0.56467
Accuracy with Scaled Data: 0.92744

Classification Report:

	precision	recall	f1-score	support
Insufficient_Weight	0.79	0.86	0.82	92
Normal_Weight	0.57	0.51	0.54	77
Obesity_Type_I	0.48	0.40	0.44	114
Obesity_Type_II	0.57	0.80	0.67	85
Obesity_Type_III	0.85	0.38	0.53	92
Overweight_Level_I	0.58	0.43	0.49	89
Overweight_Level_II	0.37	0.62	0.46	85
accuracy			0.56	634
macro avg	0.60	0.57	0.56	634
weighted avg	0.60	0.56	0.56	634

Tuning our models

- Afterwards, we decided to tune our models to see if we could get better results. We began by tuning the KNN model :

```
Fitting 5 folds for each of 528 candidates, totalling 2640 fits
Accuracy Score = 0.94
{'algorithm': 'auto', 'metric': 'manhattan', 'n_neighbors': 6, 'p': 1, 'weights': 'distance'}

Classification Report:
                    precision    recall  f1-score   support

Insufficient_Weight       0.96      0.98      0.97        92
      Normal_Weight       0.92      0.87      0.89        77
     Obesity_Type_I       0.95      0.96      0.95       114
    Obesity_Type_II       0.94      0.95      0.95        85
   Obesity_Type_III       0.99      0.98      0.98        92
 Overweight_Level_I       0.91      0.92      0.92        89
Overweight_Level_II       0.94      0.94      0.94        85

           accuracy                           0.94       634
          macro avg       0.94      0.94      0.94       634
       weighted avg       0.94      0.94      0.94       634
```

- We tuned it on the following parameters : n_neighbors, weights, metric, algorithm and the power parameter ‘p’.

Tuning our models

- Then we tuned the random forest classifier on the following parameters : n_estimators, criterion, max_depth and min_sample_leaf. On the screenshot below, we didn't use the min_sample_leaf parameter because the results were slightly worse with it instead of n_estimators (we tried using both but there are too many fits and the notebook crashed. In the end, we tried using them separately.)

```
Fitting 5 folds for each of 342 candidates, totalling 1710 fits
Accuracy Score = 0.96
{'criterion': 'gini', 'max_depth': 40, 'n_estimators': 160}

Classification Report:
                     precision    recall  f1-score   support

Insufficient_Weight       0.97      0.98      0.97        92
      Normal_Weight       0.93      0.91      0.92        77
     Obesity_Type_I       0.95      0.97      0.96       114
    Obesity_Type_II       0.98      0.98      0.98        85
   Obesity_Type_III       1.00      1.00      1.00        92
 Overweight_Level_I       0.93      0.93      0.93        89
Overweight_Level_II       0.95      0.93      0.94        85

           accuracy                           0.96       634
          macro avg       0.96      0.96      0.96       634
       weighted avg       0.96      0.96      0.96       634
```

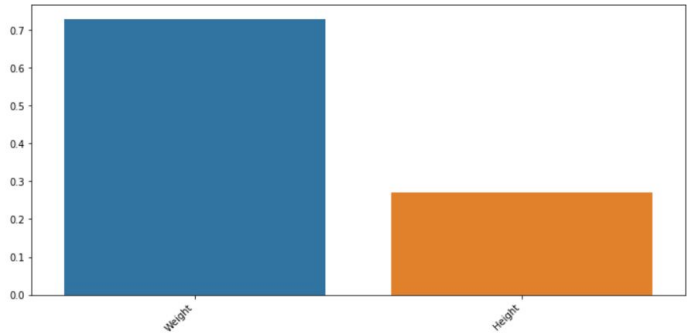
Tuning our models

- To finish our tuning, we tuned the decision tree on the following parameters: max_depth, min_samples_leaf and criterion

```
Fitting 5 folds for each of 162 candidates, totalling 810 fits
Accuracy Score = 0.92
{'criterion': 'entropy', 'max_depth': 20, 'min_samples_leaf': 10}

Classification Report:
                     precision    recall  f1-score   support

Insufficient_Weight       0.99      0.97      0.98        92
      Normal_Weight       0.88      0.83      0.85        77
     Obesity_Type_I       0.95      0.91      0.93       114
    Obesity_Type_II       0.95      0.96      0.96        85
   Obesity_Type_III       1.00      0.98      0.99        92
 Overweight_Level_I       0.79      0.91      0.84        89
Overweight_Level_II       0.88      0.85      0.86        85

           accuracy                           0.92       634
          macro avg       0.92      0.92      0.92       634
       weighted avg       0.92      0.92      0.92       634
```

Feature importance

- After tuning our models, we decided to keep the best one which is the Random Forest and figure out the importance of each feature. We then plotted the importance of each one:




```
Best Features:
Variable: Weight            Importance: 0.73
Variable: Height            Importance: 0.27
```

- The difference in importance between both variables is very significant. This can be explained by the formula of the BMI which is weight/height².

Two category target variable

- After doing all that, we decided to see if having a two category target variable was giving us better results. To do so, we had to modify the target variable a bit. We mapped the original target variable into a two category target variable: ‘Obesity’ 1 or 0.

```
# map values
weight_categories = { 'Normal_Weight':0, 'Overweight_Level_I':0,
                'Overweight_Level_II':0, 'Obesity_Type_I':1,
                'Obesity_Type_II':1, 'Obesity_Type_III':1, 'Insufficient_Weight':0}

# map values
df_prep['Obesity'] = df_prep['NObeyesdad'].map(weight_categories)
```

Result from our model

- We decided to run only a Random Forest classifier as it had the best accuracy in the prediction of the correct label of the weight category and because this part was mostly a test to see what the results would be.

```
Random Forest:
----------------------------------------------------------------
Accuracy: 0.99842
Accuracy with Scaled Data: 0.99842

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       343
           1       1.00      1.00      1.00       291

    accuracy                           1.00       634
   macro avg       1.00      1.00      1.00       634
weighted avg       1.00      1.00      1.00       634
```

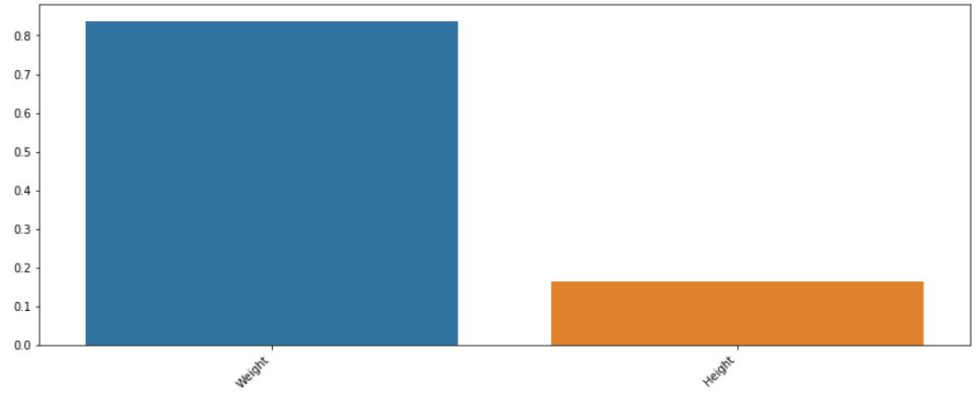
Tuning our Random Forest Classifier

- We then tuned our model according to the following parameters: criterion, max_depth and n_estimators

```
Fitting 5 folds for each of 342 candidates, totalling 1710 fits
Accuracy Score = 1.00
{'criterion': 'gini', 'max_depth': 40, 'n_estimators': 160}

Classification Report:
              precision    recall  f1-score   support

   Not Obese       1.00      1.00      1.00       343
       Obese       1.00      1.00      1.00       291

    accuracy                           1.00       634
   macro avg       1.00      1.00      1.00       634
weighted avg       1.00      1.00      1.00       634
```

Feature importance

- After witnessing the results, we tried visualizing the feature importance in this case:




```
Best Features:
Variable: Weight                Importance: 0.84
Variable: Height                Importance: 0.16
```

- We can see very clearly that the weight is much more important than the height. The difference in importance is more significant than when we tried predicting all weight labels.

Results when we kept all variables as predictors

- During our work, we tried using all variables to predict the target variable. On the next slides, we can see the results we had with this method:

Random Forest:

Accuracy: 0.91483
Accuracy with Scaled Data: 0.9164

Classification Report:

	precision	recall	f1-score	support
Insufficient_Weight	0.92	0.96	0.94	92
Normal_Weight	0.72	0.81	0.76	77
Obesity_Type_I	0.97	0.94	0.96	114
Obesity_Type_II	1.00	1.00	1.00	85
Obesity_Type_III	1.00	1.00	1.00	92
Overweight_Level_I	0.88	0.78	0.83	89
Overweight_Level_II	0.89	0.91	0.90	85
accuracy			0.91	634
macro avg	0.91	0.91	0.91	634
weighted avg	0.92	0.91	0.92	634

Random Forest Classifier

Decision Tree:

Accuracy: 0.94637
Accuracy with Scaled Data: 0.94479

Classification Report:

	precision	recall	f1-score	support
Insufficient_Weight	0.95	0.99	0.97	92
Normal_Weight	0.94	0.83	0.88	77
Obesity_Type_I	0.92	0.96	0.94	114
Obesity_Type_II	0.98	0.95	0.96	85
Obesity_Type_III	1.00	1.00	1.00	92
Overweight_Level_I	0.90	0.94	0.92	89
Overweight_Level_II	0.94	0.93	0.93	85
accuracy			0.95	634
macro avg	0.95	0.94	0.94	634
weighted avg	0.95	0.95	0.95	634

Decision Tree Classifier

KNN:

Accuracy: 0.8817
Accuracy with Scaled Data: 0.77603

Classification Report:

	precision	recall	f1-score	support
Insufficient_Weight	0.83	0.97	0.89	92
Normal_Weight	0.84	0.47	0.60	77
Obesity_Type_I	0.91	0.94	0.92	114
Obesity_Type_II	0.95	0.99	0.97	85
Obesity_Type_III	1.00	0.99	0.99	92
Overweight_Level_I	0.79	0.91	0.85	89
Overweight_Level_II	0.84	0.84	0.84	85
accuracy			0.88	634
macro avg	0.88	0.87	0.87	634
weighted avg	0.88	0.88	0.87	634

KNN

Results when we kept all variables as predictors

KNN model tuned

```
Fitting 5 folds for each of 528 candidates, totalling 2640 fits
Accuracy Score = 0.86
{'algorithm': 'auto', 'metric': 'manhattan', 'n_neighbors': 4, 'p': 1, 'weights': 'distance'}

Classification Report:
                     precision    recall  f1-score   support

Insufficient_Weight       0.86      0.93      0.90        92
      Normal_Weight       0.70      0.70      0.70        77
     Obesity_Type_I       0.91      0.84      0.88       114
    Obesity_Type_II       0.90      0.99      0.94        85
   Obesity_Type_III       0.99      0.98      0.98        92
 Overweight_Level_I       0.81      0.72      0.76        89
Overweight_Level_II       0.78      0.81      0.79        85

           accuracy                           0.86       634
          macro avg       0.85      0.85      0.85       634
       weighted avg       0.86      0.86      0.86       634
```

Random Forest model tuned

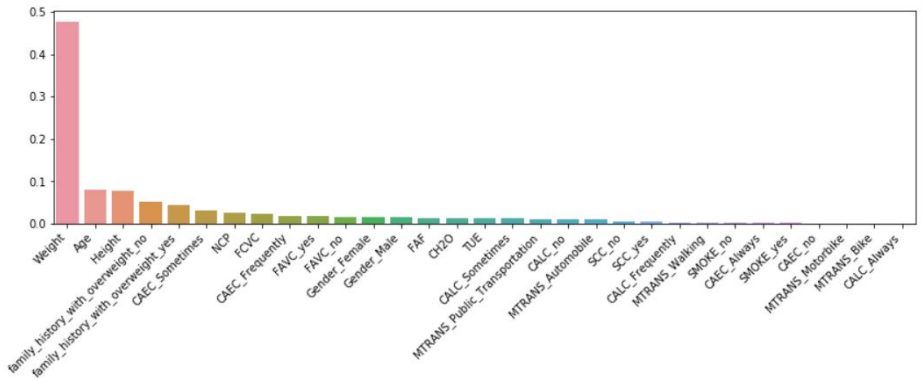
```
Fitting 5 folds for each of 342 candidates, totalling 1710 fit
Accuracy Score = 0.93
{'criterion': 'entropy', 'max_depth': 30, 'n_estimators': 170}

Classification Report:
                     precision    recall  f1-score   support

Insufficient_Weight       0.93      0.96      0.94        92
      Normal_Weight       0.76      0.83      0.80        77
     Obesity_Type_I       0.99      0.94      0.96       114
    Obesity_Type_II       0.99      1.00      0.99        85
   Obesity_Type_III       1.00      1.00      1.00        92
 Overweight_Level_I       0.89      0.81      0.85        89
Overweight_Level_II       0.90      0.93      0.91        85

           accuracy                           0.93       634
          macro avg       0.92      0.92      0.92       634
       weighted avg       0.93      0.93      0.93       634
```

Decision Tree model tuned

```
Fitting 5 folds for each of 162 candidates, totalling 810 fits
Accuracy Score = 0.92
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 10}

Classification Report:
                     precision    recall  f1-score   support

Insufficient_Weight       0.98      0.97      0.97        92
      Normal_Weight       0.88      0.82      0.85        77
     Obesity_Type_I       0.95      0.89      0.92       114
    Obesity_Type_II       0.98      0.98      0.98        85
   Obesity_Type_III       1.00      1.00      1.00        92
 Overweight_Level_I       0.83      0.92      0.87        89
Overweight_Level_II       0.84      0.88      0.86        85

           accuracy                           0.92       634
          macro avg       0.92      0.92      0.92       634
       weighted avg       0.92      0.92      0.92       634
```

Results when we kept all variables as predictors

- We also displayed the importance of each feature as well as its importance

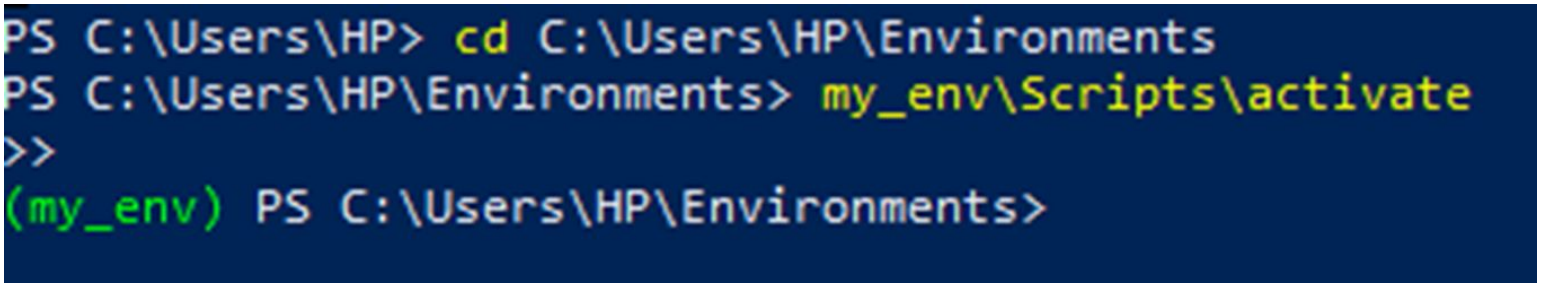



```
Best Features:
Variable: Weight               Importance: 0.48
Variable: Age                  Importance: 0.08
Variable: Height               Importance: 0.08
Variable: family_history_with_overweight_no Importance: 0.05
Variable: family_history_with_overweight_yes Importance: 0.04
Variable: CAEC_Sometimes       Importance: 0.03
Variable: FCVC                 Importance: 0.02
Variable: NCP                  Importance: 0.02
Variable: Gender_Female        Importance: 0.02
Variable: FAVC_no              Importance: 0.02
```

In the end, the weight is the most important feature in all cases which makes sense because of its strong impact in the calculus of the BMI.

Transformation of the model into an API of our choice

- At first we created a virtual environment but we had few problems and we found an easier way which didn't make you install anything.



- So, instead of the virtual environment with the html and css files, we went with only a python file.