

# PYTHON FOR DATA ANALYSIS OBESITY DATA SET

PUJOL Corentin – PELET Quentin

# TABLE OF CONTENTS

- I. Presentation of the dataset
- II. Visualization of the data
- III. Processing models of Machine Learning
- IV. Flask API

# PRESENTATION OF THE DATASET



# PRESENTATION OF THE DATASET

Our dataset include data for the estimation of obesity levels in individuals from the countries of Mexico, Peru and Colombia, with ages between 14 and 61 and based on their eating habits and physical condition.

<b>Data Set Characteristics:</b>	Multivariate	<b>Number of Instances:</b>	2111	<b>Area:</b>	Life
<b>Attribute Characteristics:</b>	Integer	<b>Number of Attributes:</b>	17	<b>Date Donated</b>	2019-08-27
<b>Associated Tasks:</b>	Classification, Regression, Clustering	<b>Missing Values?</b>	N/A	<b>Number of Web Hits:</b>	66392

The data contains 17 attributes and 2111 records, the records are labeled with the class variable NObesity (Obesity Level), that allows classification of the data using the values of Insufficient Weight, Normal Weight, Overweight Level I, Overweight Level II, Obesity Type I, Obesity Type II and Obesity Type III.

Finally, 77% of the data was generated synthetically using the Weka tool and the SMOTE filter, 23% of the data was collected directly from users through a web platform.

# PRESENTATION OF THE DATASET

The attributes related with eating habits are:

- Frequent consumption of high caloric food (FAVC)
- Frequency of consumption of vegetables (FCVC)
- Number of main meals (NCP)
- Consumption of food between meals (CAEC)
- Consumption of water daily (CH20)
- Consumption of alcohol (CALC)

The attributes related with the physical condition are:

- Calories consumption monitoring (SCC)
- Physical activity frequency (FAF)
- Time using technology devices (TUE)
- Transportation used (MTRANS)
- Other variables obtained were: Gender, Age, Height and Weight.

# PRESENTATION OF THE DATASET

The data was collected through a survey, which included the following questions:

What is your gender?	•Female •Male
what is your age?	Numeric value
what is your height?	Numeric value in meters
what is your weight?	Numeric value in kilograms
Has a family member suffered or suffers from overweight?	•Yes •No
Do you eat high caloric food frequently?	•Yes •No
Do you usually eat vegetables in your meals?	•Never •Sometimes •Always
How many main meals do you have daily?	•Between 1 y 2 •Three •More than three
Do you eat any food between meals?	•No •Sometimes •Frequently •Always

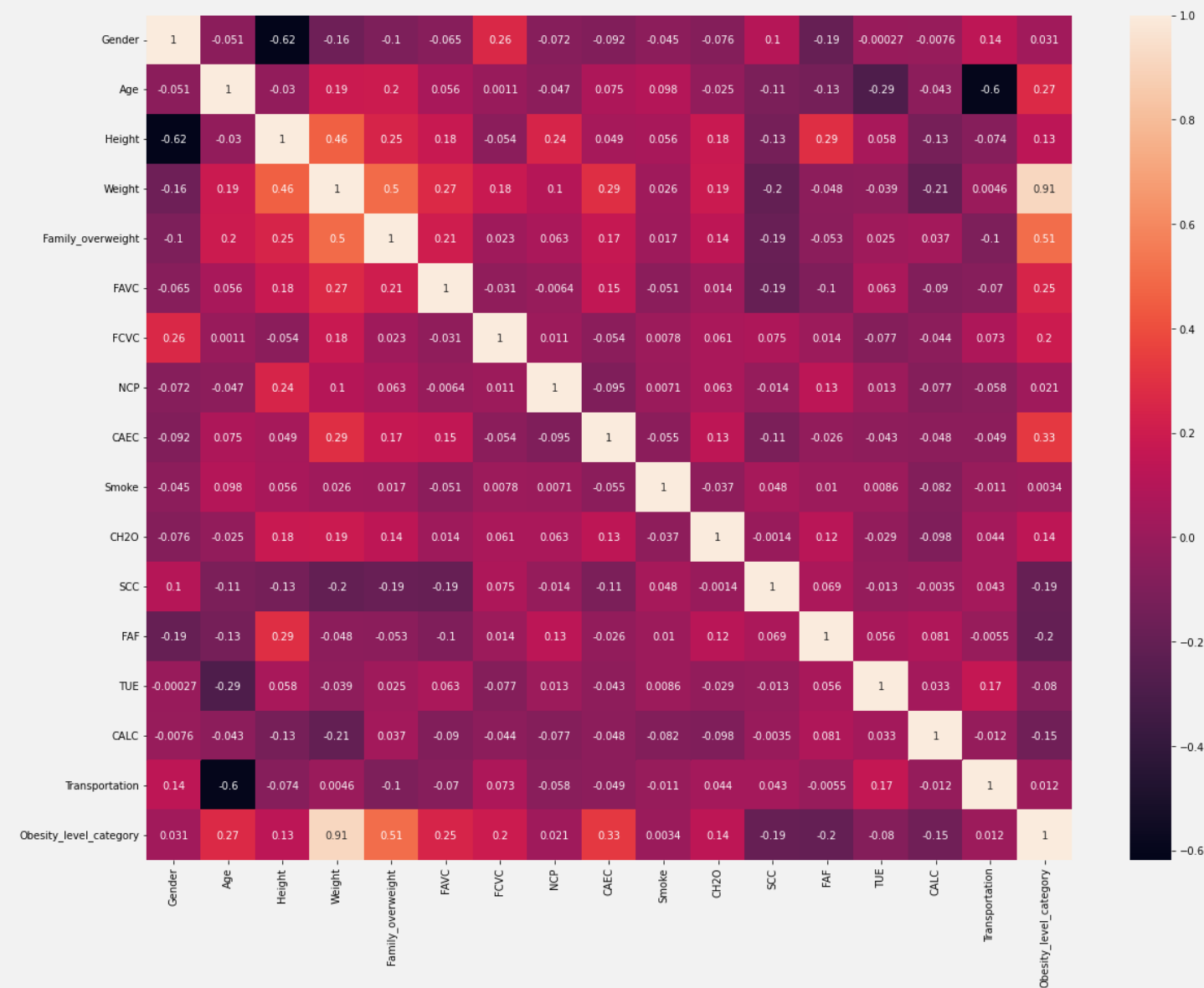
Do you smoke?	•Yes •No
How much water do you drink daily?	•Less than a liter •Between 1 and 2 L •More than 2 L
Do you monitor the calories you eat daily?	•Yes •No
How often do you have physical activity?	•I do not have •1 or 2 days •2 or 4 days •4 or 5 days
How much time do you use technological devices	•0–2 hours •3–5 hours •More than 5 hours
how often do you drink alcohol?	•I do not drink •Sometimes •Frequently •Always
Which transportation do you usually use?	•Automobile •Motorbike •Bike •Public Transportation •Walking

# VISUALIZATION OF THE DATA





# Correlation matrix

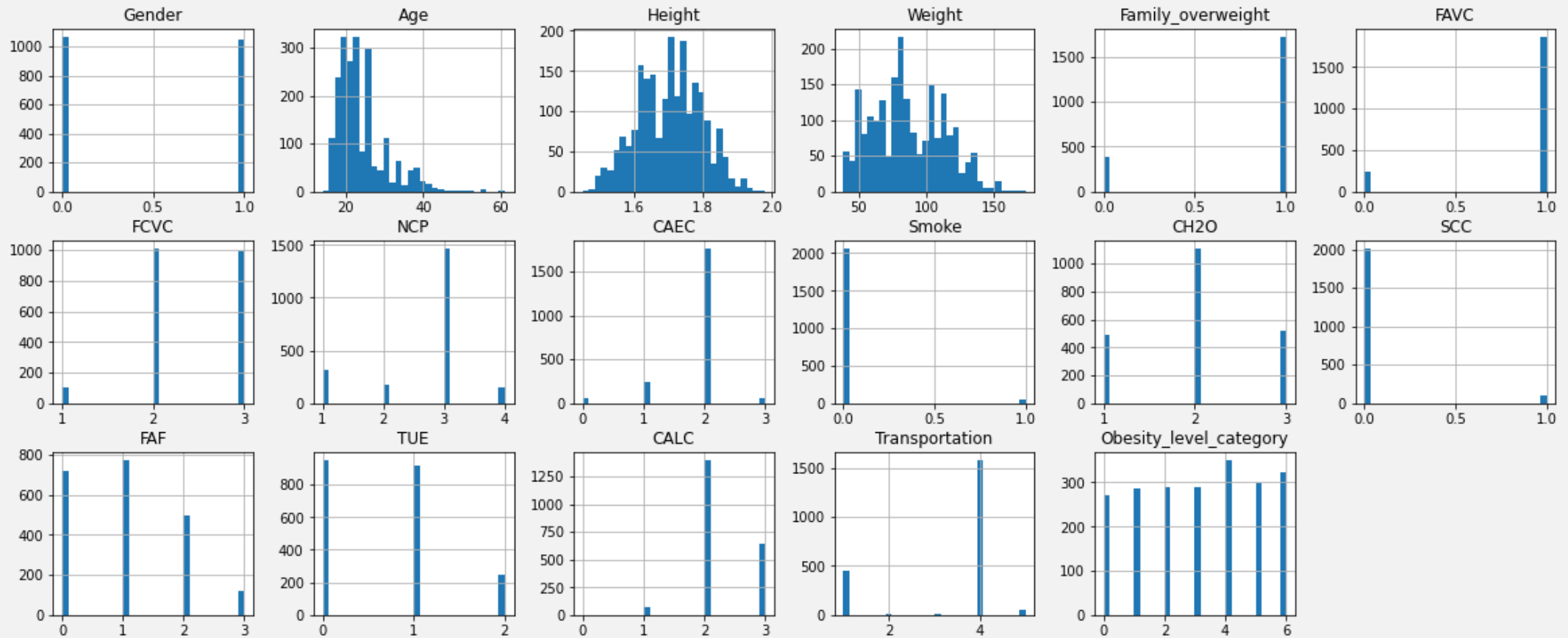


The variable Weight is highly correlated with the level of obesity, which is logic since weight is very important in determining the level of obesity with the BMI

Some variables have very little influence on the level of obesity, which is the case for example with the variables Smoke, Gender, NCP, or even the variable Transportation

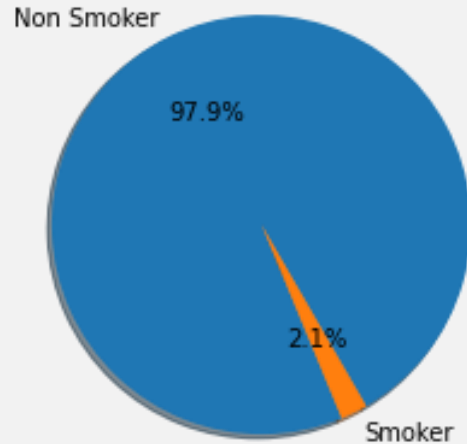


# Distribution of the data



The age of the population studied is quite young. While height and weight are well distributed. So we have a very diverse population to study.

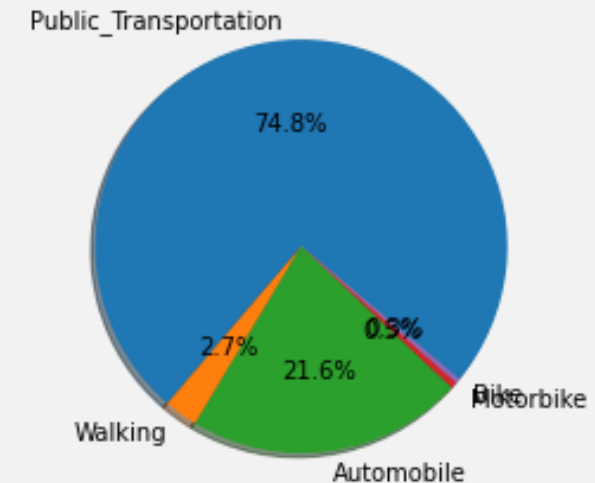
## Smoke variable



In our population, we have only 2% of smokers, which is not very representative of a real population

For example in Mexico, we find that 16% of the population smokes

## Transportation variable



This variable contains values for the modes of transport of the population. They are either physical (Bike, Walking) or non-physical (Automobile, Motorbike, Public\_Transportation)

In order for this variable to have a greater impact on the prediction, we have chosen to transform into 0 the modes of transport that do not require physical effort, and into 1 the others.

# PROCESSING MODELS OF MACHINE LEARNING



## Choice of the models

We chose 5 different models to predict our data:

- Linear Discriminant Analysis
- K-Nearest Neighbors
- Support Vector Machine
- Random Forest
- Gradient Boosting Classifier

We set up the models by finding the best parameters

We used them to predict either 7 classes or 2

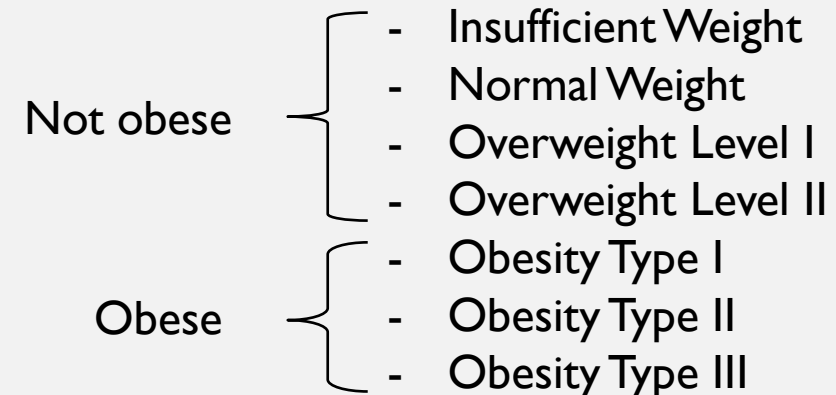
## Prediction of all classes

We predict the 7 classes that were present in the dataset:

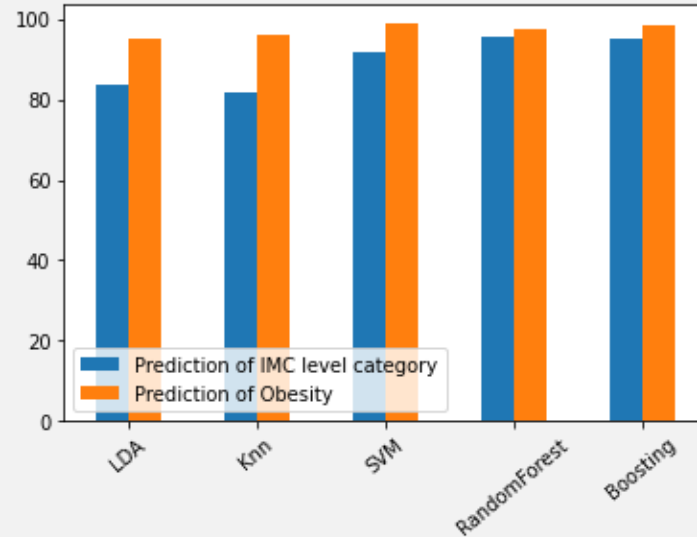
- Insufficient Weight
- Normal Weight
- Overweight Level I
- Overweight Level II
- Obesity Type I
- Obesity Type II
- Obesity Type III

## Binary classification

We wanted to simplify the prediction by reducing the number of classes to be predicted to 2:



Performance of the predictions of the different models implemented



## Comparison of the models

- On these visualisations, we can see that the binary prediction has a better score for each model.
- Moreover, we observe that the two best models are Boosting and Random Forest, although SVM has a better score for binary classification

	LDA	Knn	SVM	RandomForest	Boosting
Prediction of IMC level category	83.93	81.92	91.97	95.84	95.12
Prediction of Obesity	95.12	96.27	99.00	97.56	98.71



# FLASK API

```

mirror_mod = modifier_ob.
set mirror object to mirror
mirror_mod.mirror_object

operation == "MIRROR_X":
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
operation == "MIRROR_Y":
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True

#selection at the end -add
modifier_ob.select= 1
modifier_ob.select=1
context.scene.objects.active
("Selected" + str(modifier_ob.select))
mirror_ob.select = 0
= bpy.context.selected_objects
data.objects[one.name].select

print("please select exactly one mirror")

-- OPERATOR CLASSES -----

types.Operator):
X mirror to the selected
object.mirror_mirror_x"
mirror X"

context):
context.active_object is not

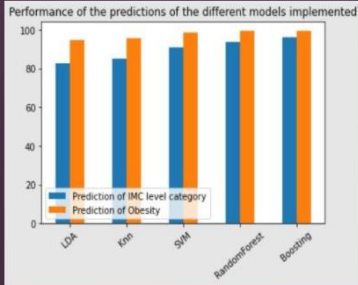
```

# FLASK API

Welcome to the API of PELET Quentin & PUJOL Corentin for predicting obesity levels

Below are two options, both of which are predictive methods. One only determines whether a person is obese or not, while the other determines whether a person falls into a BMI range

Which model do you want to choose?



Performance of the predictions of the different models implemented

Model	Prediction of IMC level category (%)	Prediction of Obesity (%)
LDA	82	95
Knn	85	95
SVM	90	98
RandomForest	95	98
Boosting	95	98

--Please choose an option--

Prediction of Obesity

--Please choose an option--

Prediction of IMC level category

This is the API home page. There are several options such as choosing the prediction model, and then two buttons to choose the type of prediction we want to get.



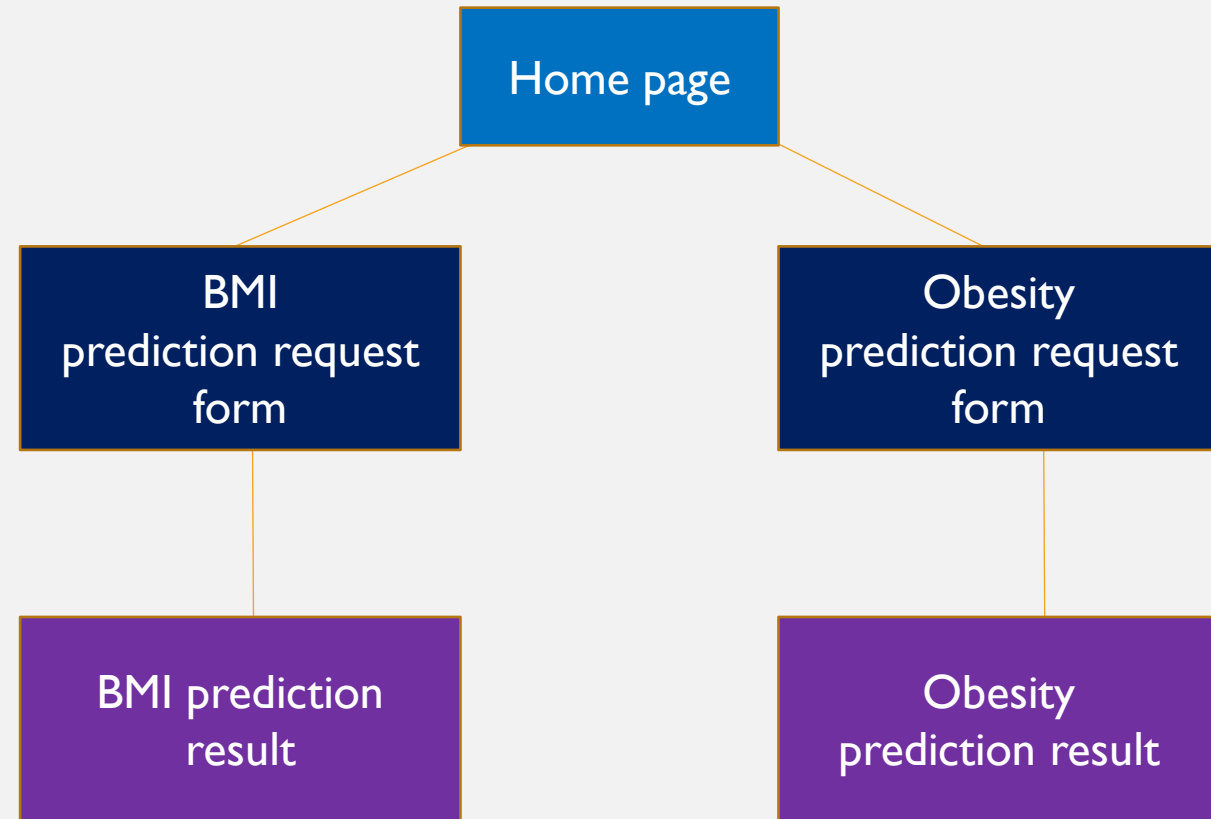
# FLASK API

Our application consists of 5 pages:

The home page, which allows you to choose which model to use and which prediction we want to make (whether a person is obese or not, or whether we want to obtain directly a BMI range for the individual).

Two pages displaying a form to be filled in with information about the different variables in our dataset (one for each type of prediction)

Two pages displaying the predictions made by our algorithms (with the possibility of filling in the form again)



# FLASK API

This page is composed of a Flask structure:

```
@app.route('/')  
def home():  
    return render_template("index_accueil.html", url='models.JPG')
```

This method, using the `render_template` librairie read our html page "index\_accueil" which is stored in the content folder in our collaboratory virtual machine.

Here is an overview of the html page on the following slide.

# FLASK API

The page style is defined in the header, and the style.css file is placed in a static folder in the same directory as the html template.

The different elements making up the page are placed in a class container division in order to facilitate the positioning of the elements.

The url\_for method is function in the Flask flask. url\_for generates a URL to an endpoint using the method passed in as an argument.

```
<!DOCTYPE html>
<html >
<head>
<meta charset="utf-8">
<title>PELET Quentin & PUJOL Corentin Machine Learning API</title>
<link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
<div>
<h1>Welcome to the API of PELET Quentin & PUJOL Corentin for predicting obesity levels</h1>
<br>
<p>Below are two options, both of which are predictive methods. One only determines whether a person is obese<br>
<label for="model">Which model do you want to choose?</label>
<br>
<br>

<br>
<div class="flex-container">
<div>
<form action="{{ url_for('Binary_prediction') }}" method="post">
<select name="model" id="model" required="required">
<option value="">--Please choose an option--</option>
<option value="LDA">LDA</option>
<option value="Knn">Knn</option>
<option value="SVM">SVM</option>
<option value="RandomForest">RandomForest</option>
<option value="Boosting">Boosting</option>
</select>
<br>
<br>
<button type="submit" class="btn btn-primary btn-large">Prediction of Obesity</button>
</form>
</div>
</div>
```

# FLASK API

The action defined by the `url_for`, will therefore read this part of the code, allowing the opening of a new page with another html template.

We retrieve the information chosen and given by the user using the `request.form.values()` function.

We place the output object afterwards in order to display the different results (predictions or possible announcement of a chosen model)

```
@app.route('/Binary_prediction',methods=['POST'])
def Binary_prediction():

    input = [x for x in request.form.values()]
    if(input[0]=="LDA"):
        my_best_model_bin=model_lg
    else:
        if(input[0]=="Knn"):
            my_best_model_bin=model_knn_bin
        else:
            if(input[0]=="SVM"):
                my_best_model_bin=model_svm_bin
            else:
                if(input[0]=="Boosting"):
                    my_best_model_bin=model_boost_bin

    return render_template("binary_classification.html", output="Vous avez choisi le modèle " + input[0])
```

LIEN GITHUB



<https://github.com/corentin-pujol/q-Obesity-types-prediction>