Python for Data Analysis

Thibaud Trarieux DIA7

Summary :

- - Dataset presentation and problematic
- - Treatment of data
- - Machine learning
- - Django API
- - Conclusion

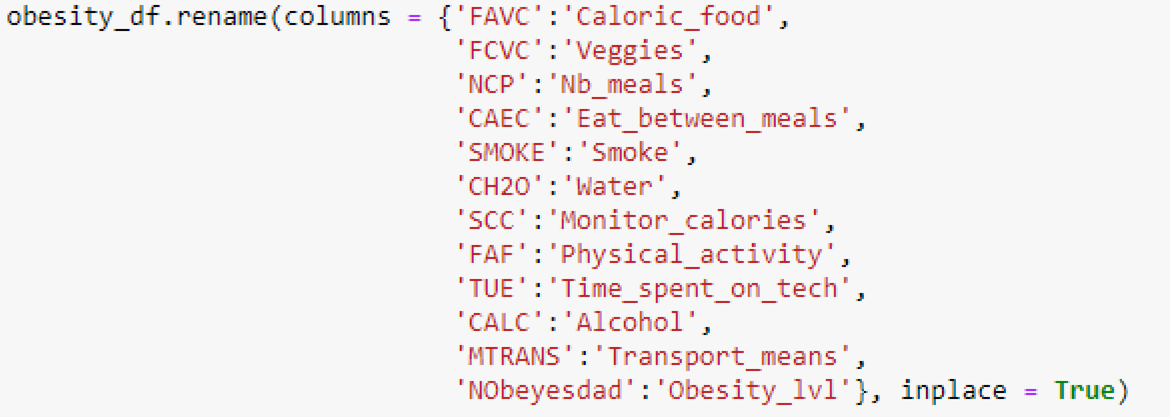
Dataset presentation

- The dataset contains 17 attributes and 2111 records, the records are labeled with the class variable Obesity_lvl, which can have the following values :
- - Insufficient Weight
- - Normal Weight
- - Overweight Level I
- - Overweight Level II
- - Obesity Type I
- - Obesity Type II
- -Obesity Type III.
- Problematic : Can we predict someone’s obesity level using the dataset ?

Treatment of data

Rename columns

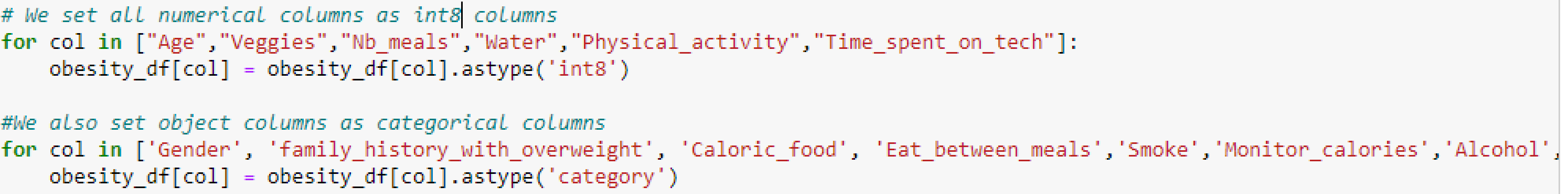
- Firstly, let’s rename columns with clearer names.



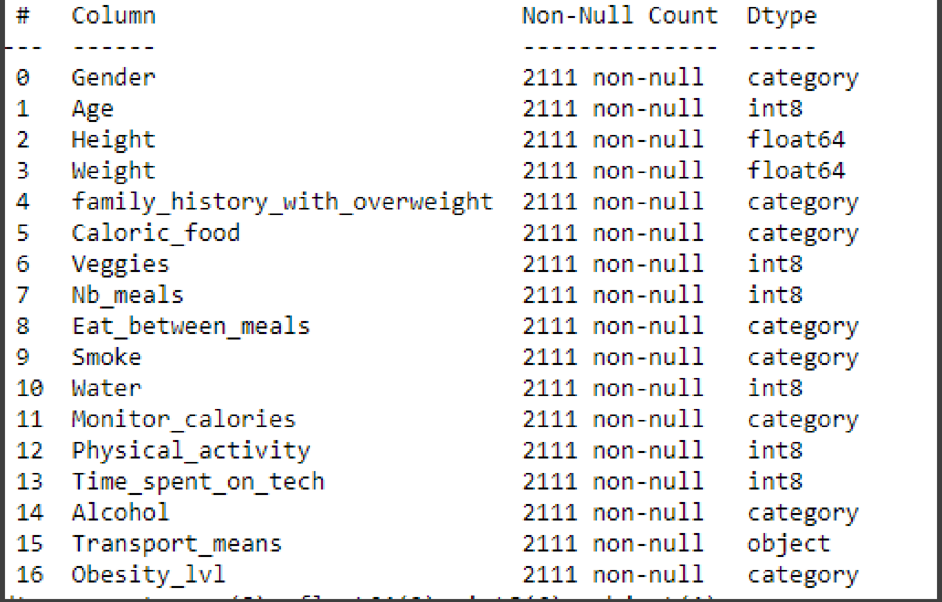
```
obesity_df.rename(columns = {'FAVC':'Caloric_food',
                             'FCVC':'Veggies',
                             'NCP':'Nb_meals',
                             'CAEC':'Eat_between_meals',
                             'SMOKE':'Smoke',
                             'CH2O':'Water',
                             'SCC':'Monitor_calories',
                             'FAF':'Physical_activity',
                             'TUE':'Time_spent_on_tech',
                             'CALC':'Alcohol',
                             'MTRANS':'Transport_means',
                             'NObeyesdad':'Obesity_lvl'}, inplace = True)
```

Changing data type

- Now, let’s set all numerical columns as int type columns and all object columns as category type columns.



```
# We set all numerical columns as int8 columns
for col in ["Age","Veggies","Nb_meals","Water","Physical_activity","Time_spent_on_tech"]:
    obesity_df[col] = obesity_df[col].astype('int8')

#We also set object columns as categorical columns
for col in ['Gender', 'family_history_with_overweight', 'Caloric_food', 'Eat_between_meals','Smoke','Monitor_calories','Alcohol',
    obesity_df[col] = obesity_df[col].astype('category')
```



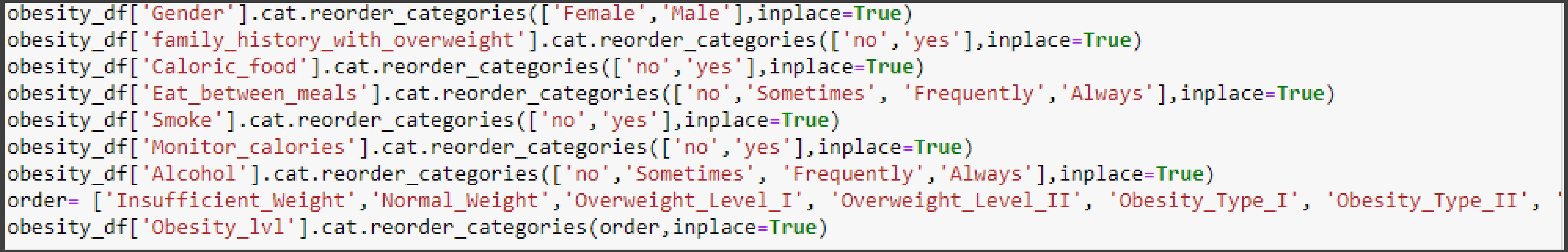
```
#   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Gender                          2111 non-null   category
 1   Age                             2111 non-null   int8
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   family_history_with_overweight  2111 non-null   category
 5   Caloric_food                    2111 non-null   category
 6   Veggies                         2111 non-null   int8
 7   Nb_meals                        2111 non-null   int8
 8   Eat_between_meals               2111 non-null   category
 9   Smoke                           2111 non-null   category
 10  Water                           2111 non-null   int8
 11  Monitor_calories                2111 non-null   category
 12  Physical_activity               2111 non-null   int8
 13  Time_spent_on_tech              2111 non-null   int8
 14  Alcohol                         2111 non-null   category
 15  Transport_means                 2111 non-null   object
 16  Obesity_lvl                     2111 non-null   category
```

For now, we keep some columns with category type because we first want to make some visualizations in the notebook.

Changing categorical data into numerical data

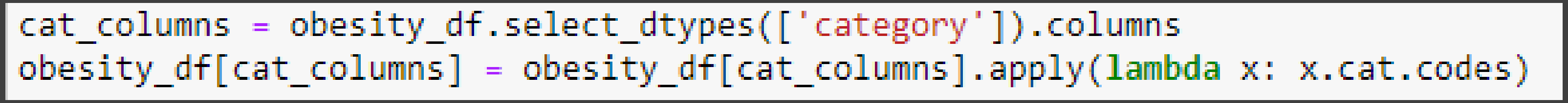
- Once we are done with visualizations, we can change categorical data into numerical data.

- Firstly, we set an order for categorical columns using ‘.cat.reorder_categories’.



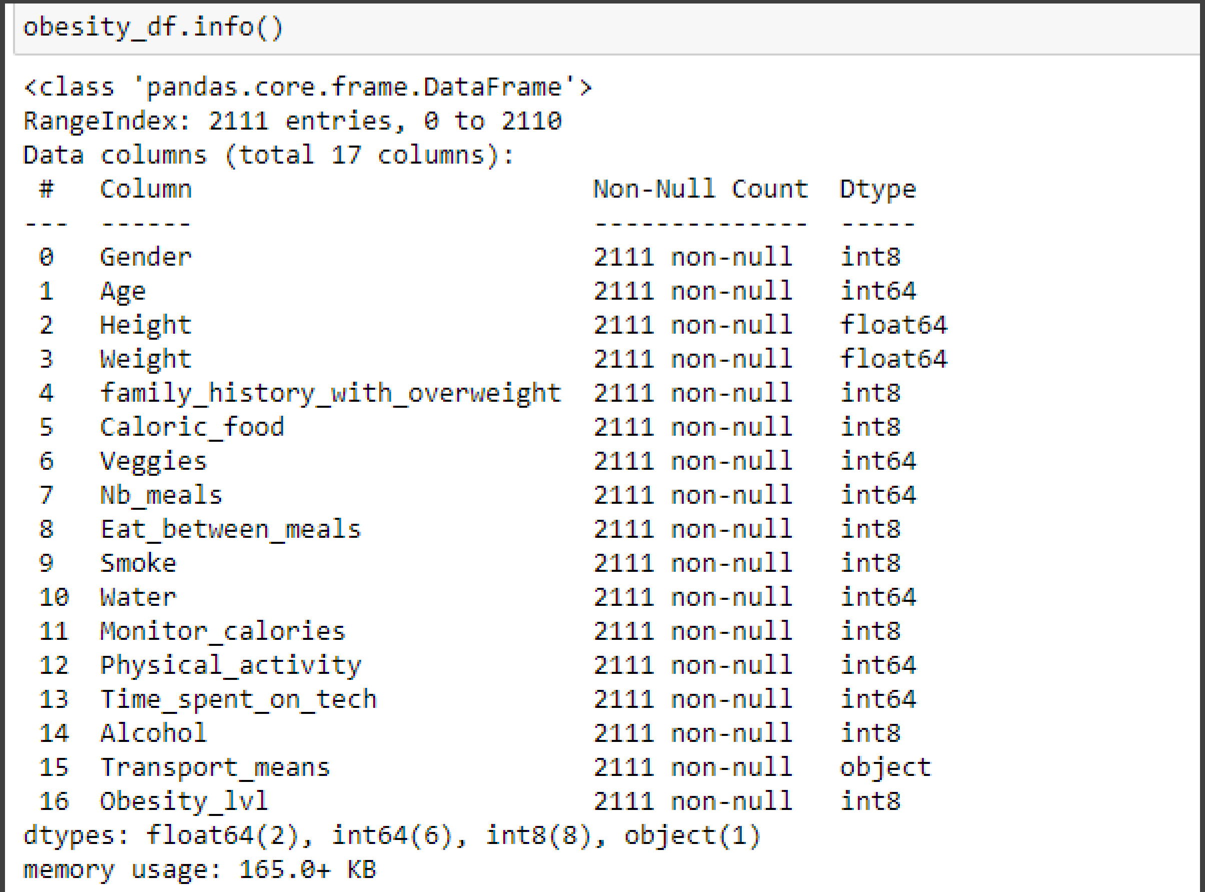
```
obesity_df['Gender'].cat.reorder_categories(['Female','Male'],inplace=True)
obesity_df['family_history_with_overweight'].cat.reorder_categories(['no','yes'],inplace=True)
obesity_df['Caloric_food'].cat.reorder_categories(['no','yes'],inplace=True)
obesity_df['Eat_between_meals'].cat.reorder_categories(['no','Sometimes', 'Frequently','Always'],inplace=True)
obesity_df['Smoke'].cat.reorder_categories(['no','yes'],inplace=True)
obesity_df['Monitor_calories'].cat.reorder_categories(['no','yes'],inplace=True)
obesity_df['Alcohol'].cat.reorder_categories(['no','Sometimes', 'Frequently','Always'],inplace=True)
order= ['Insufficient_Weight','Normal_Weight','Overweight_Level_I', 'Overweight_Level_II', 'Obesity_Type_I', 'Obesity_Type_II', '
obesity_df['Obesity_lvl'].cat.reorder_categories(order,inplace=True)
```

- Once values of each categorical column are ordered, we can change it into numeric data.



```
cat_columns = obesity_df.select_dtypes(['category']).columns
obesity_df[cat_columns] = obesity_df[cat_columns].apply(lambda x: x.cat.codes)
```

- Now, all the variables are numerical, except the Transport_means column but we will deal later with it.

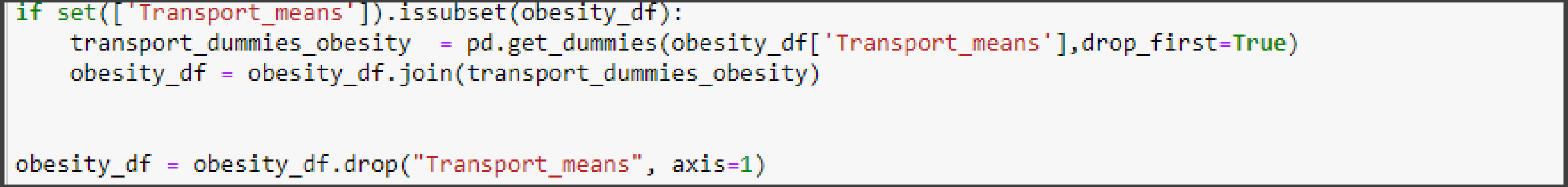


```
obesity_df.info()
```

```
<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   int8
 1   Age                             2111 non-null   int64
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   family_history_with_overweight  2111 non-null   int8
 5   Caloric_food                    2111 non-null   int8
 6   Veggies                         2111 non-null   int64
 7   Nb_meals                        2111 non-null   int64
 8   Eat_between_meals               2111 non-null   int8
 9   Smoke                           2111 non-null   int8
 10  Water                           2111 non-null   int64
 11  Monitor_calories                2111 non-null   int8
 12  Physical_activity               2111 non-null   int64
 13  Time_spent_on_tech              2111 non-null   int64
 14  Alcohol                         2111 non-null   int8
 15  Transport_means                 2111 non-null   object
 16  Obesity_lvl                     2111 non-null   int8
dtypes: float64(2), int64(6), int8(8), object(1)
memory usage: 165.0+ KB
```

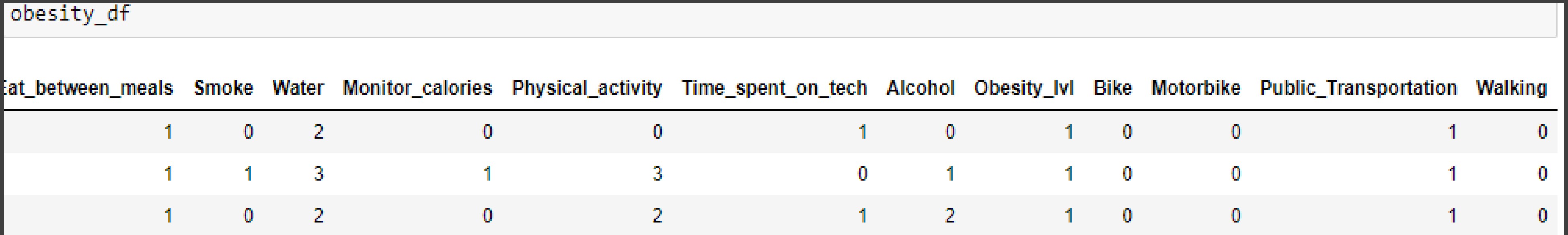
Transport_means column & Creation of new variables

- Values of Transport_means columns were not orderable. So, I modified the structure of data and create new columns using ‘pd.get_dummies’.
- Then, we can drop Transport_means column without loosing any information.



```
if set(['Transport_means']).issubset(obesity_df):
    transport_dummies_obesity  = pd.get_dummies(obesity_df['Transport_means'],drop_first=True)
    obesity_df = obesity_df.join(transport_dummies_obesity)


obesity_df = obesity_df.drop("Transport_means", axis=1)
```

- We have now 4 new columns : Bike, Motorbike, Public_Transportation, Walking.



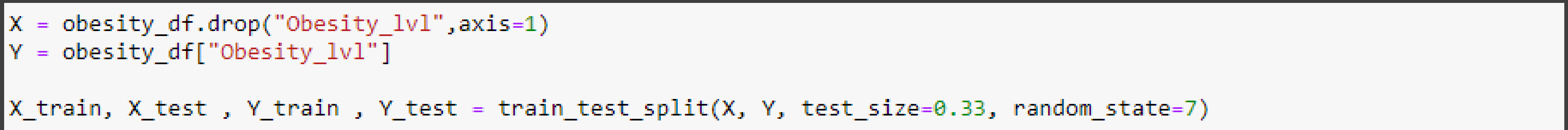
```
obesity_df
```

Eat_between_meals	Smoke	Water	Monitor_calories	Physical_activity	Time_spent_on_tech	Alcohol	Obesity_lvl	Bike	Motorbike	Public_Transportation	Walking
1	0	2	0	0	1	0	1	0	0	1	0
1	1	3	1	3	0	1	1	0	0	1	0
1	0	2	0	2	1	2	1	0	0	1	0

Machine learning

Split of the dataset

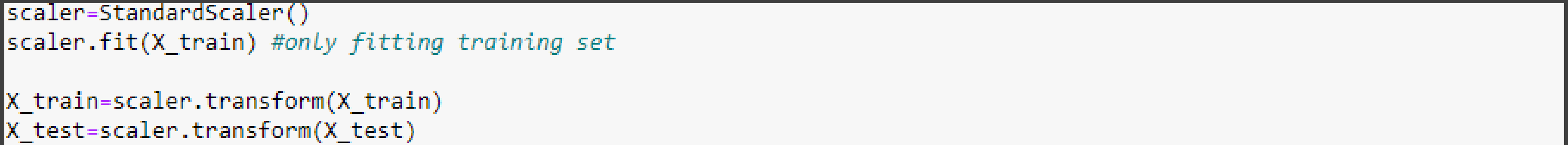
- Firstly, let's split data into training set and test set. 66% of data will be used to train the model and 33% to test the model.



```
X = obesity_df.drop("Obesity_lvl",axis=1)
Y = obesity_df["Obesity_lvl"]

X_train, X_test , Y_train , Y_test = train_test_split(X, Y, test_size=0.33, random_state=7)
```

Fitting data

Let’s fit training set to improve the accuracy of our machine learning algorithms.

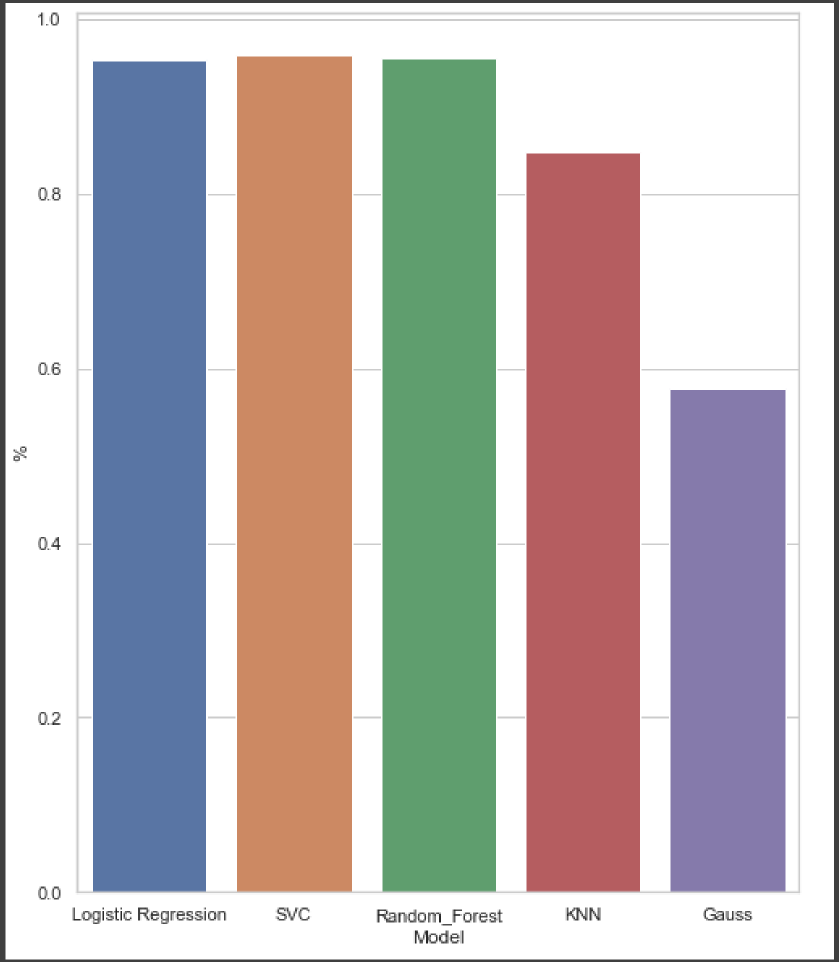


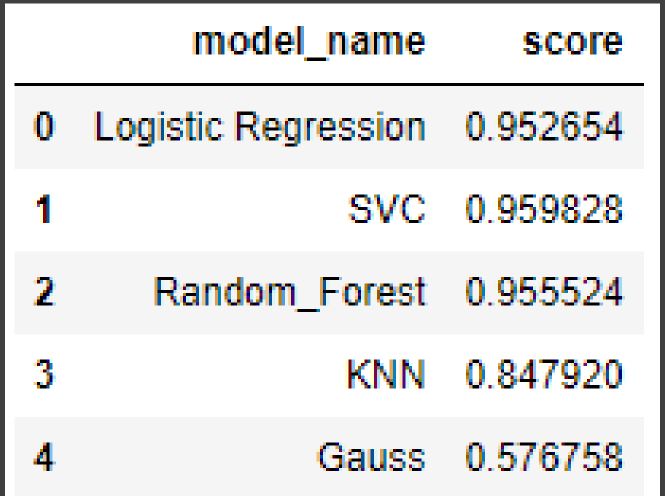
```
scaler=StandardScaler()
scaler.fit(X_train) #only fitting training set

X_train=scaler.transform(X_train)
X_test=scaler.transform(X_test)
```

Algorithms & GridSearch

- We will use the following algorithms from sckit-learn library :
- Logistic Regression
- Support Vector Machines
- Random Forest
- KNeighborsClassifier
- Gaussian Naive Bayes
- For each model, we firstly use a grid seach to find the best hyperparameters. Then, we run each model with the best hyperparameters found. Finally, we add to score_models the score of each model.

Results of



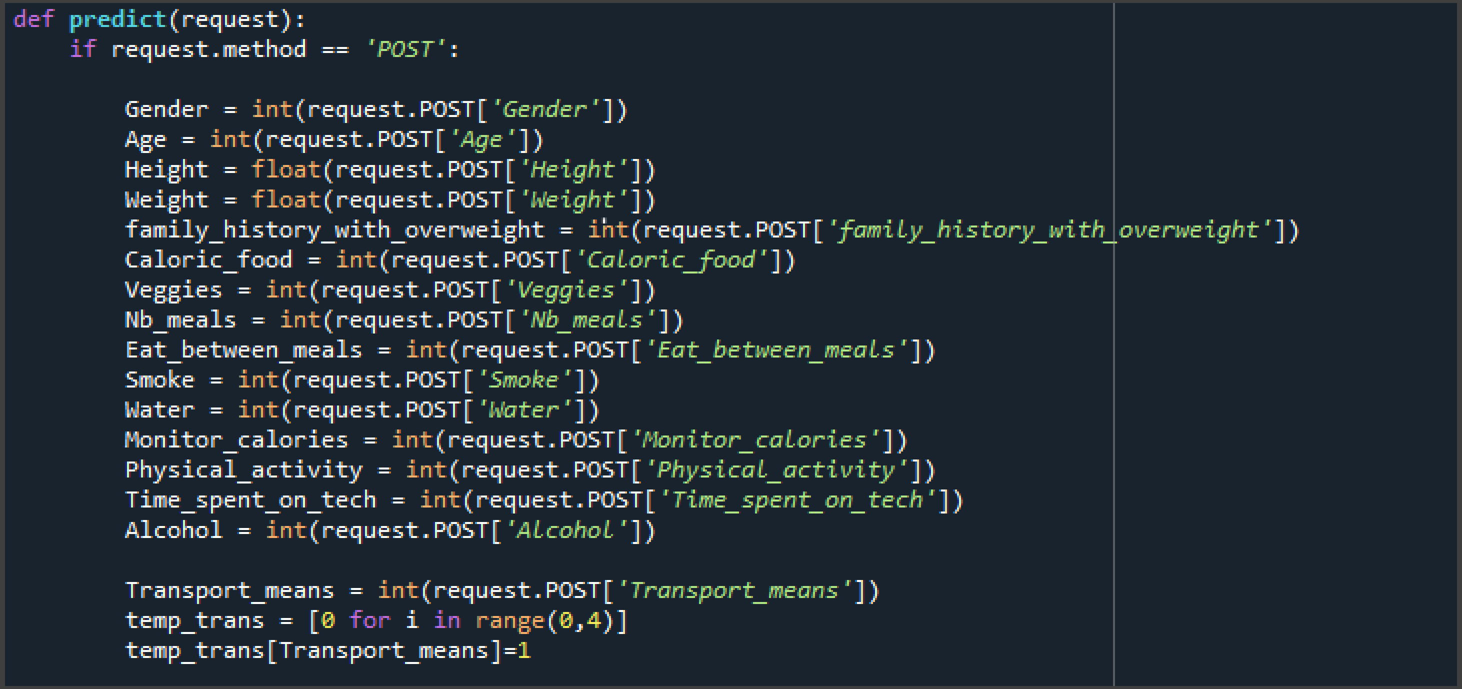


	model_name	score
0	Logistic Regression	0.952654
1	SVC	0.959828
2	Random_Forest	0.955524
3	KNN	0.847920
4	Gauss	0.576758

We notice that 3 models hit a really good score (SVC, Random Forest and Logistic Regression). The best model seems to be SVC with 0.959828 of accuracy.

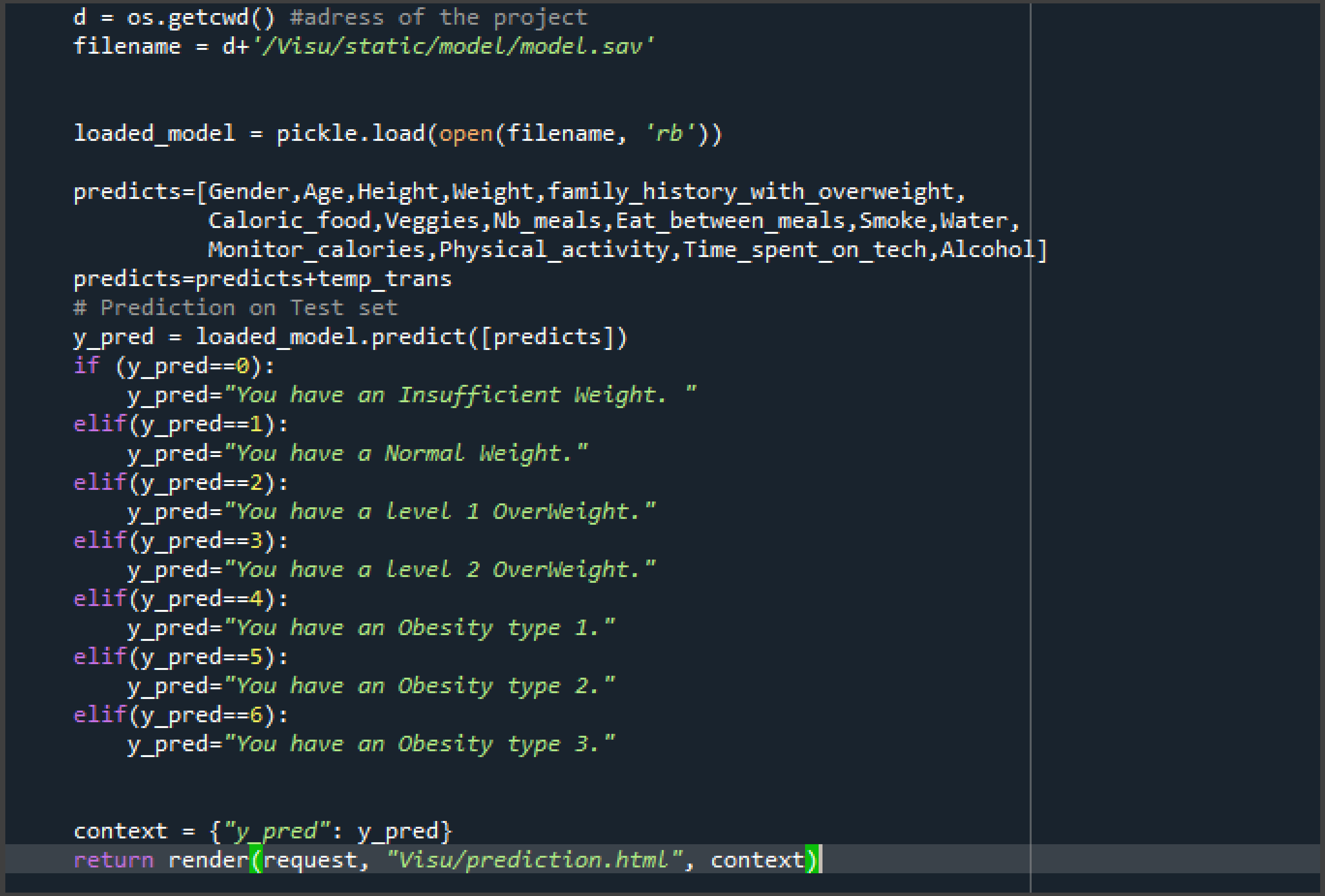
API Django

- I decided to use Django to make an API. How does it work ?
- Basically, we firstly we take data the user will post on the api.



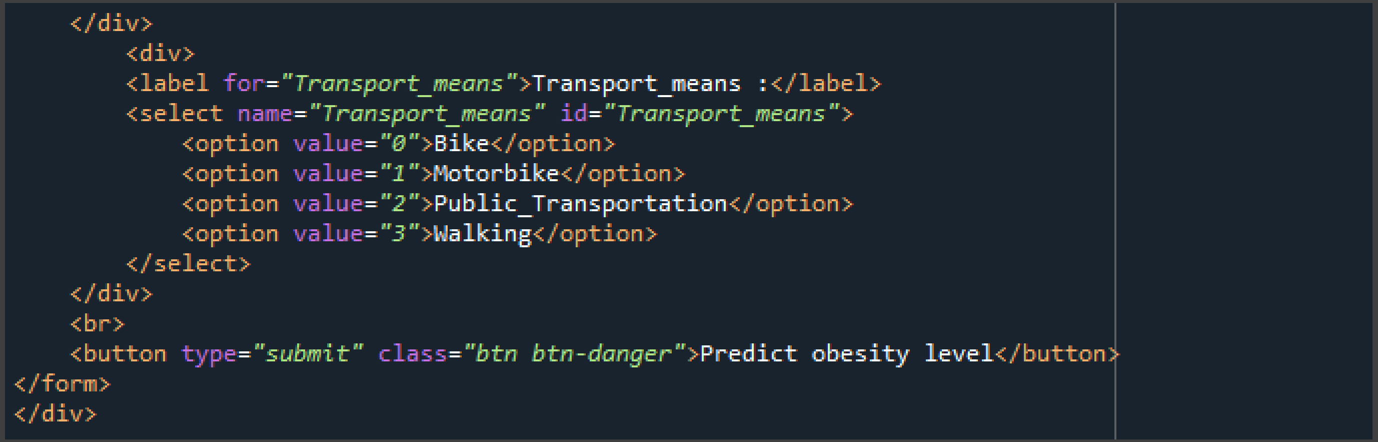
```
def predict(request):
    if request.method == 'POST':

        Gender = int(request.POST['Gender'])
        Age = int(request.POST['Age'])
        Height = float(request.POST['Height'])
        Weight = float(request.POST['Weight'])
        family_history_with_overweight = int(request.POST['family_history_with_overweight'])
        Caloric_food = int(request.POST['Caloric_food'])
        Veggies = int(request.POST['Veggies'])
        Nb_meals = int(request.POST['Nb_meals'])
        Eat_between_meals = int(request.POST['Eat_between_meals'])
        Smoke = int(request.POST['Smoke'])
        Water = int(request.POST['Water'])
        Monitor_calories = int(request.POST['Monitor_calories'])
        Physical_activity = int(request.POST['Physical_activity'])
        Time_spent_on_tech = int(request.POST['Time_spent_on_tech'])
        Alcohol = int(request.POST['Alcohol'])

        Transport_means = int(request.POST['Transport_means'])
        temp_trans = [0 for i in range(0,4)]
        temp_trans[Transport_means]=1
```

- Then, we call the best model we found previously from the notebook. (SCV)
- And we make the predictions on the test set.



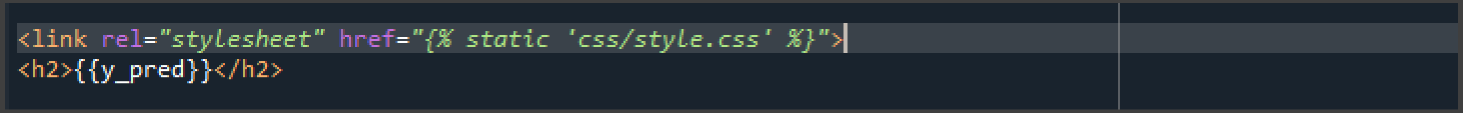
```
d = os.getcwd() #adress of the project
filename = d+'/Visu/static/model/model.sav'


loaded_model = pickle.load(open(filename, 'rb'))

predicts=[Gender,Age,Height,Weight,family_history_with_overweight,
          Caloric_food,Veggies,Nb_meals,Eat_between_meals,Smoke,Water,
          Monitor_calories,Physical_activity,Time_spent_on_tech,Alcohol]
predicts=predicts+temp_trans
# Prediction on Test set
y_pred = loaded_model.predict([predicts])
if (y_pred==0):
    y_pred="You have an Insufficient Weight. "
elif(y_pred==1):
    y_pred="You have a Normal Weight."
elif(y_pred==2):
    y_pred="You have a level 1 OverWeight."
elif(y_pred==3):
    y_pred="You have a level 2 OverWeight."
elif(y_pred==4):
    y_pred="You have an Obesity type 1."
elif(y_pred==5):
    y_pred="You have an Obesity type 2."
elif(y_pred==6):
    y_pred="You have an Obesity type 3."


context = {"y_pred": y_pred}
return render(request, "Visu/prediction.html", context)
```

- After that we must do 2 html pages. One that allows the user to enter data(index.html and one to show the results (prediction.html).



```
    </div>
        <div>
        <label for="Transport_means">Transport_means :</label>
        <select name="Transport_means" id="Transport_means">
            <option value="0">Bike</option>
            <option value="1">Motorbike</option>
            <option value="2">Public_Transportation</option>
            <option value="3">Walking</option>
        </select>
    </div>
    <br>
    <button type="submit" class="btn btn-danger">Predict obesity level</button>
</form>
</div>
```

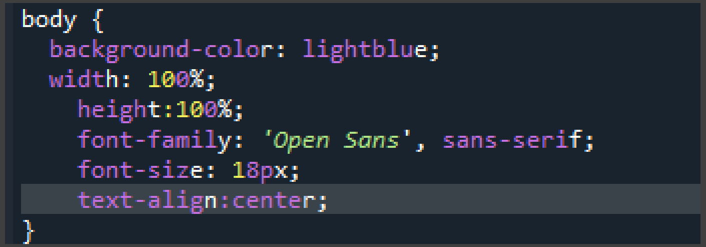
Piece of index.html



```
<link rel="stylesheet" href="{% static 'css/style.css' %}">
<h2>{{y_pred}}</h2>
```

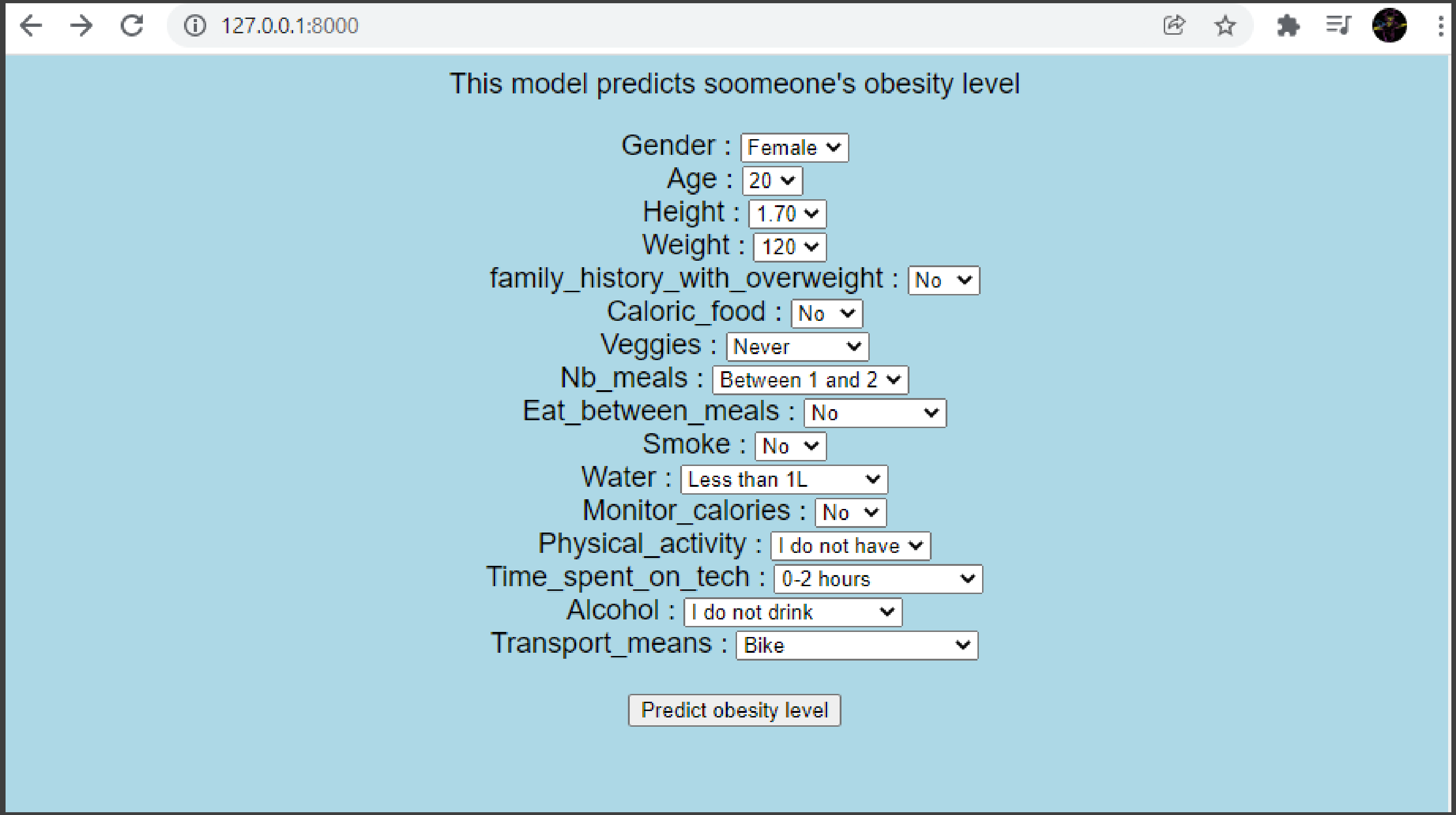
Prediction.html

- Finally, we add a style.css file to make the web page look nicer.



```
body {
  background-color: lightblue;
  width: 100%;
    height:100%;
    font-family: 'Open Sans', sans-serif;
    font-size: 18px;
    text-align:center;
}
```

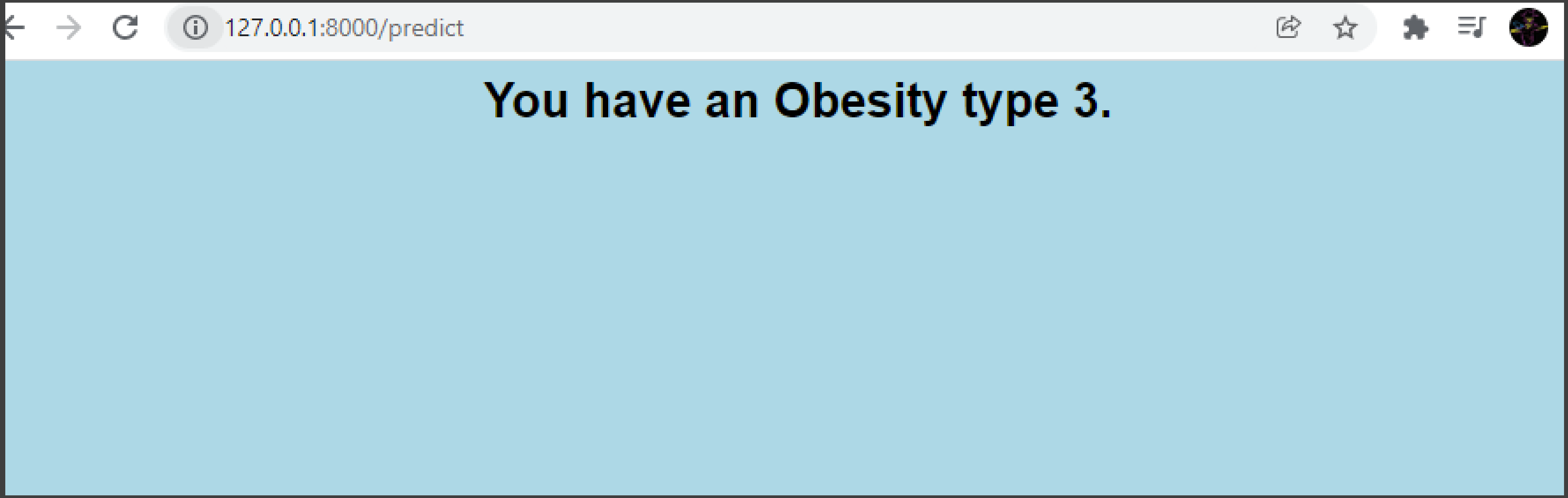
Webpage 1



This model predicts soomeone's obesity level

Gender : Female
Age : 20
Height : 1.70
Weight : 120
family_history_with_overweight : No
Caloric_food : No
Veggies : Never
Nb_meals : Between 1 and 2
Eat_between_meals : No
Smoke : No
Water : Less than 1L
Monitor_calories : No
Physical_activity : I do not have
Time_spent_on_tech : 0-2 hours
Alcohol : I do not drink
Transport_means : Bike

Webpage 2



You have an Obesity type 3.

Conclusion

- Finally, the best machine learning algorithm to predict someone’s obesity level is SVC, with 0.959828 of accuracy.
- Random forest, logistic regression and Knn also had a great score.
- However, Gaussian algorithm was not performant, with 0.576758 of accuracy.