Python for Data analysis

Arnaud MAGARIAN – ESILV A5

Python for Data Analysis ML – final project

Arnaud MAGARIAN

Dataset

Estimation of obesity levels based on eating habits and physical condition

Dataset.

Number of instances: 2111

Number of Attributes: 17

- Number of features: 16
- Number of target labels: 1

Missing values: None

This dataset include data for the estimation of obesity levels in individuals from the countries of Mexico, Peru and Colombia, based on their eating habits and physical condition. 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. 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.

Dataset link.

Attribute information link.

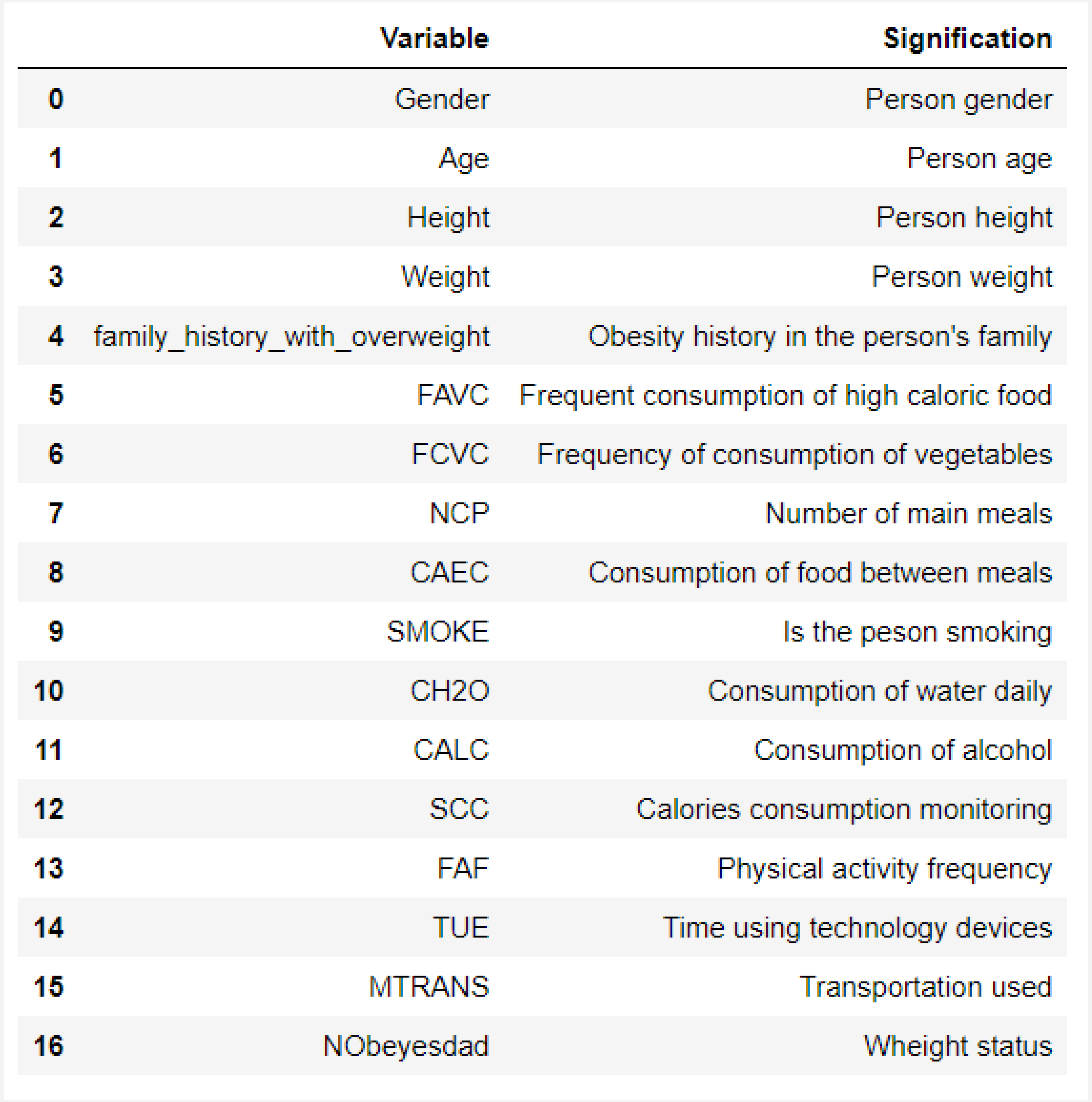
THE GOAL

The goal of this dataset is to be able to predict a person’s weight status using the 16 features on the right, Nobeyesdad being the weight status.

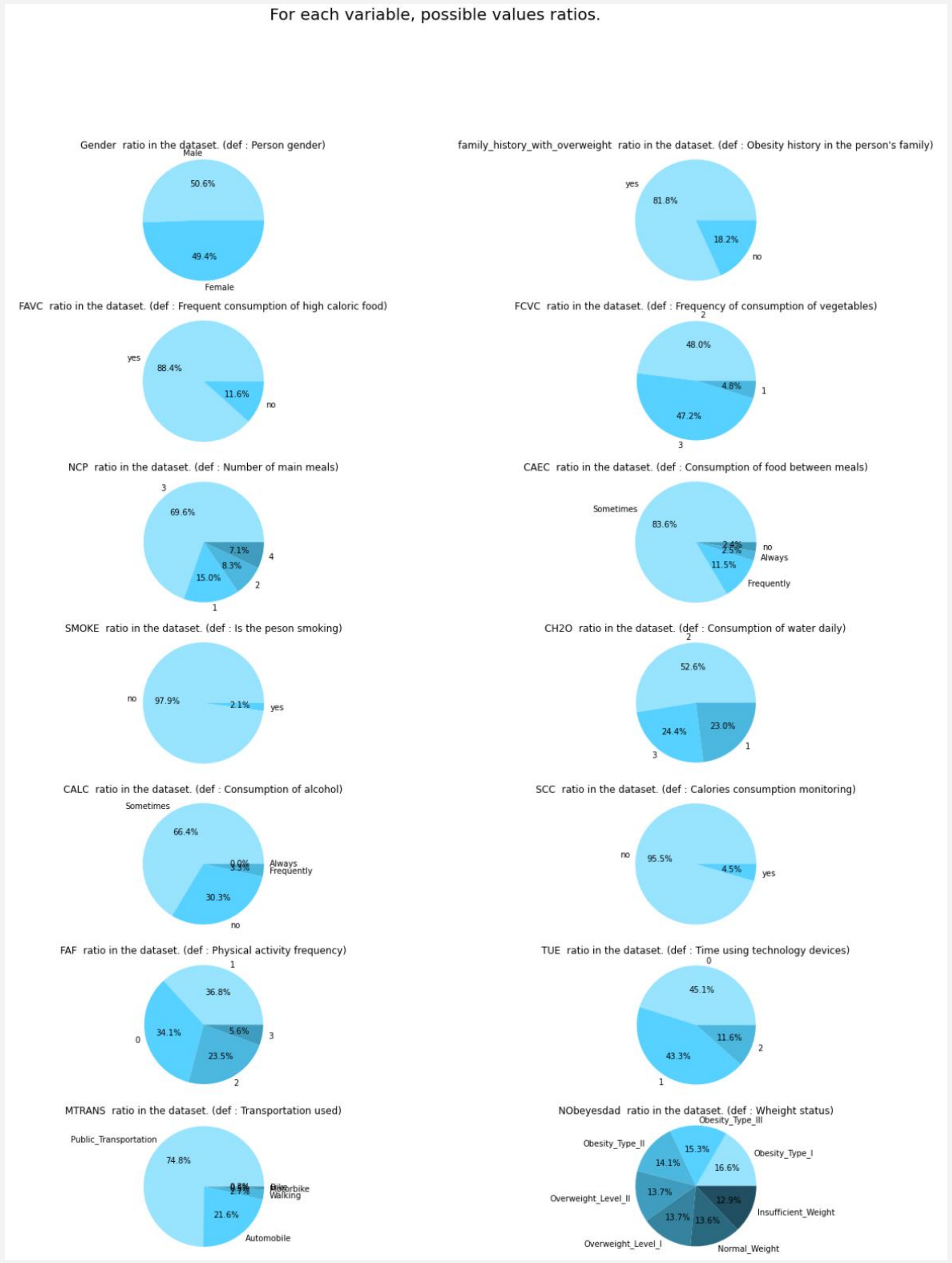
The dataset is a mix of qualitative and quantitative features.

Qualitative :
- Gender
- family_history_with_overweight
- FAVC
- CAEC
- SMOKE
- SCC
- CALC
- MTRANS

Quantitative :
- Age
- Weight
- Height
- FCVC
- NCP
- CH2O
- FAF
- TUE



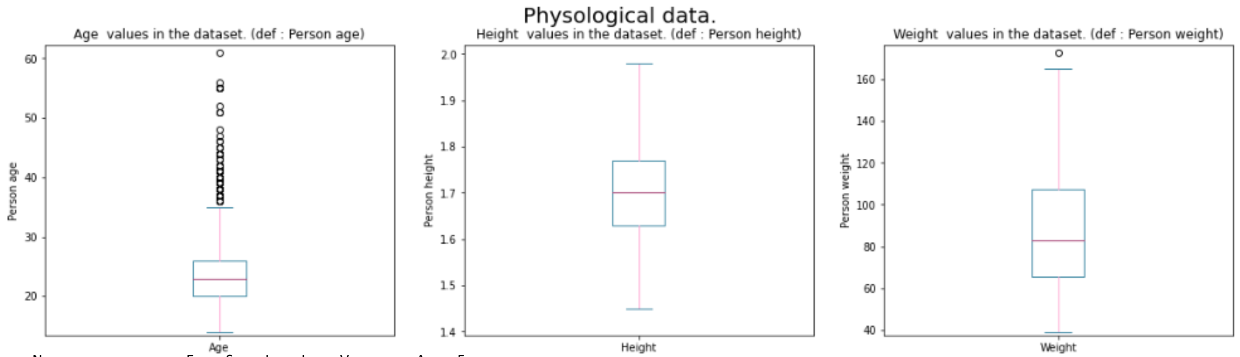
	Variable	Signification
0	Gender	Person gender
1	Age	Person age
2	Height	Person height
3	Weight	Person weight
4	family_history_with_overweight	Obesity history in the person's family
5	FAVC	Frequent consumption of high caloric food
6	FCVC	Frequency of consumption of vegetables
7	NCP	Number of main meals
8	CAEC	Consumption of food between meals
9	SMOKE	Is the peson smoking
10	CH2O	Consumption of water daily
11	CALC	Consumption of alcohol
12	SCC	Calories consumption monitoring
13	FAF	Physical activity frequency
14	TUE	Time using technology devices
15	MTRANS	Transportation used
16	NObeyesdad	Wheight status



DATA REPARTITION

The above function displays a pie chart for each variable having a fix set of possible values. Such charts only serve the purpose of better understanding the dataset population. As shown by the above, the characteristics defining the population and the weight (object of the study) are well balanced (gender and weight status --> first and last pie charts) The others should be the differentiating parameters when training our model in identifying obesity based on a set of parameters.

Those three bow plots show the data repartition of the physiological attributes. They are all well distributed. In the case of the age, the dataset population is quite young even if we have a bunch of outliers. Height and weight are well distributed.



DATA INTERPRETATION

The pie charts and scatter plots we obtained are both great. On one hand, the scatter plots allow us to compare the weight status to more features, giving us more complete graphs. On the other hand, pie charts are very easy to read and interpret with giving us a more precise reading (percentage). In the end, both plot types are useful to cross information and better understand the dataset before doing any machine learning.

Let's now interpret all the above charts. I am going to go row by row :

1. Gender:
Normal weight, overweight & obesity: gender does not seem to be related to those variables.
Insufficient weight: it seems that females are more likely to have an insufficient weight than males.

2. Obesity history in the person's family:
Having an obesity historic in the person's family seems to influence towards normal weight or insufficient weight.
On the other hand, the large majority of overweight and obese people do not have any obesity history in their family.

3. Frequent consumption of high caloric food:
Intuitive results : Heavier is the person, higher is his high caloric food consumption frequency.
Insufficient weight people also consume more caloric food than people with a normal weight.

4. Frequency of consumption of vegetables:
Does not seems to be such a differentiating factor. Obese people seems to be the one with the highest frequency of vegetables consumption.

5. Number of main meals:
The large majority (+70%) of obese and normal weight people eat 3 main meals per day.
26.5% of the people having an insufficient weight eat 4 main meals per day.

DATA INTERPRETATION

6. Consumption of food between meals:

96.1% of obese people and 87.2% of overweight people consume sometimes food between meals.

28.9% of normal weight people and 44.5% of insufficient weight people consume frequently food between meals.

7. Smoking:

Just a really small number of smokers in the dataset.

8. Consumption of water daily:

Quite even, independent of the weight status. Overweight and obese people tend to consume a bit more water than people with a normal or insufficient way.

9. Consumption of alcohol:

Heavier the person, higher is the consumption of alcohol.

10. Calories consumption monitoring:

Higher the weight, less likely is the person to monitor its calories consumption.

11. Physical activity frequency:

Higher the weight, lower the physical activity frequency.

12. Time using technology devices:

Does not seems to be link to the weight.

13. Transportation used:

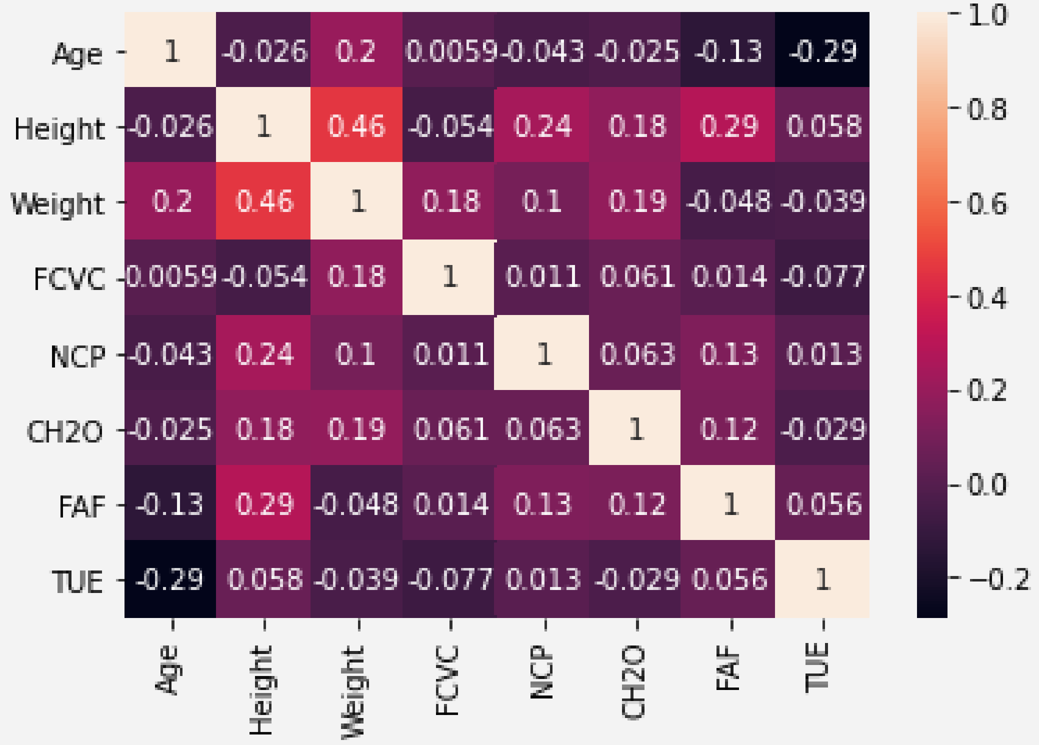
69.1% or more people use public transportation.

Higher the weight higher is the use of Automobile as a transportation mean.

11.1% of people with a normal weight walk. 2.2% or less of the rest walk.

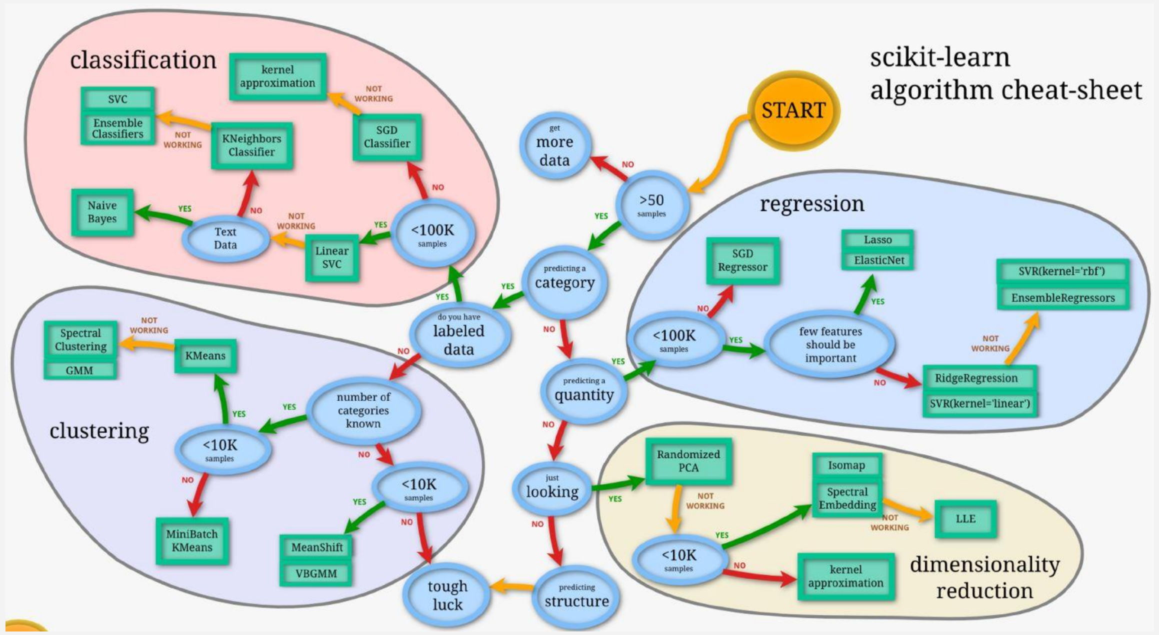
DATA CORRELATION

As shown by the heatmap on the left, the dataset features are not really correlated (in appearance) as the highest value we have is 0,46 and the next one drops to 0,29.
Moreover, as a lot of our data is text we can't display all features using a heatmap.



	Age	Height	Weight	FCVC	NCP	CH2O	FAF	TUE
Age	1	-0.026	0.2	0.0059	-0.043	-0.025	-0.13	-0.29
Height	-0.026	1	0.46	-0.054	0.24	0.18	0.29	0.058
Weight	0.2	0.46	1	0.18	0.1	0.19	-0.048	-0.039
FCVC	0.0059	-0.054	0.18	1	0.011	0.061	0.014	-0.077
NCP	-0.043	0.24	0.1	0.011	1	0.063	0.13	0.013
CH2O	-0.025	0.18	0.19	0.061	0.063	1	0.12	-0.029
FAF	-0.13	0.29	-0.048	0.014	0.13	0.12	1	0.056
TUE	-0.29	0.058	-0.039	-0.077	0.013	-0.029	0.056	1

MACHINE LEARNING ALGORITHM



The goal of using Machine Learning on this dataset is to determine the weight state of a person. Basically, we want to classify them into weight classes (overweight, normal weight, etc.). So we should use classification algorithms;

To identify which scikit-learn algorithm could correspond, I will use the following "cheat_sheet" helping to better understand which algorithm is suited to which dataset.

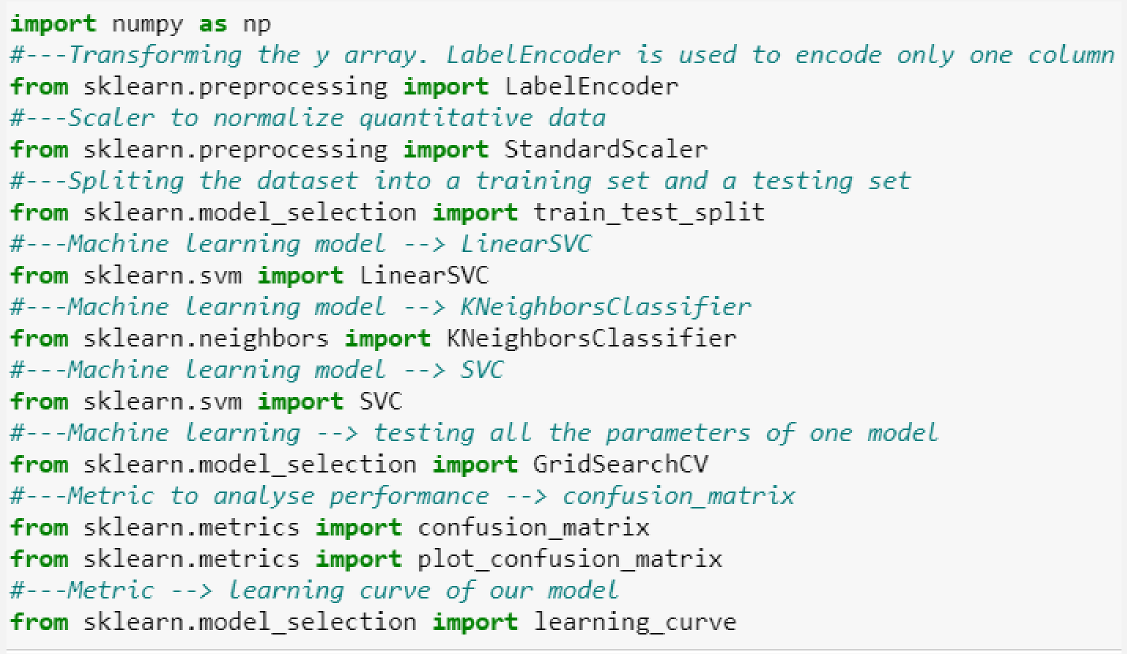
Using my dataset :

- >50 samples --> Yes
- Predicting a category --> Yes
- Labeled data --> Yes, I now know that I should use classification algorithms.
    - <100K samples --> Yes
    - Linear SVC
        - Working well --> Done!
        - Text Data
            - Yes --> Naive Bayes
            - No --> KNeighbors Classifier
                - Not working --> SVC / Ensemble classifiers

Both what I know and the cheat-sheet are pointing towards a classification problem. KNeighbors are not use in the case of text data. But as I will encode my data, we will be able to use this algorithm as well as SVC and EnsembleClassifiers.

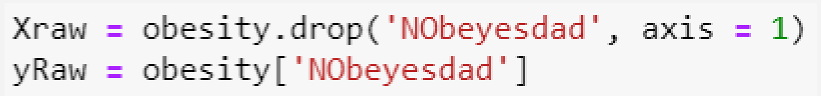
LIBRARIES

On the left, the list of libraries that were used for the preprocessing and the machine learning.

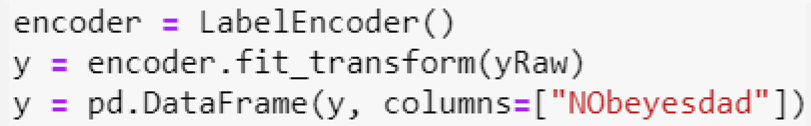


```python
import numpy as np
#---Transforming the y array. LabelEncoder is used to encode only one column
from sklearn.preprocessing import LabelEncoder
#---Scaler to normalize quantitative data
from sklearn.preprocessing import StandardScaler
#---Spliting the dataset into a training set and a testing set
from sklearn.model_selection import train_test_split
#---Machine learning model --> LinearSVC
from sklearn.svm import LinearSVC
#---Machine learning model --> KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier
#---Machine learning model --> SVC
from sklearn.svm import SVC
#---Machine learning --> testing all the parameters of one model
from sklearn.model_selection import GridSearchCV
#---Metric to analyse performance --> confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
#---Metric --> learning curve of our model
from sklearn.model_selection import learning_curve
```

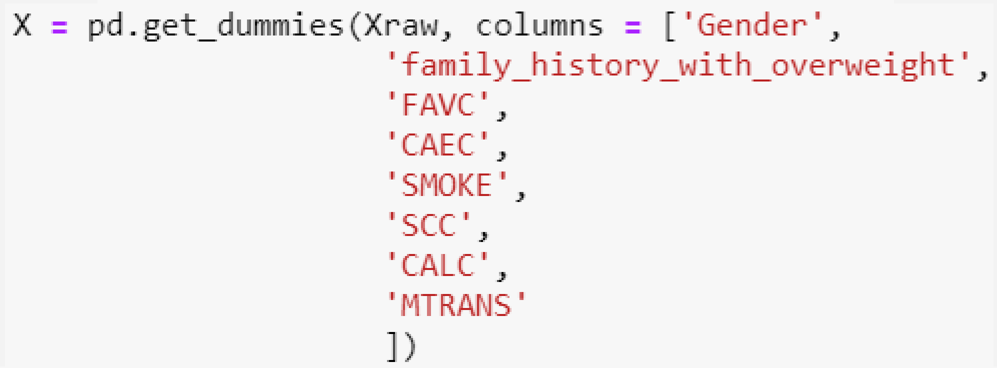
PREPROCESSING



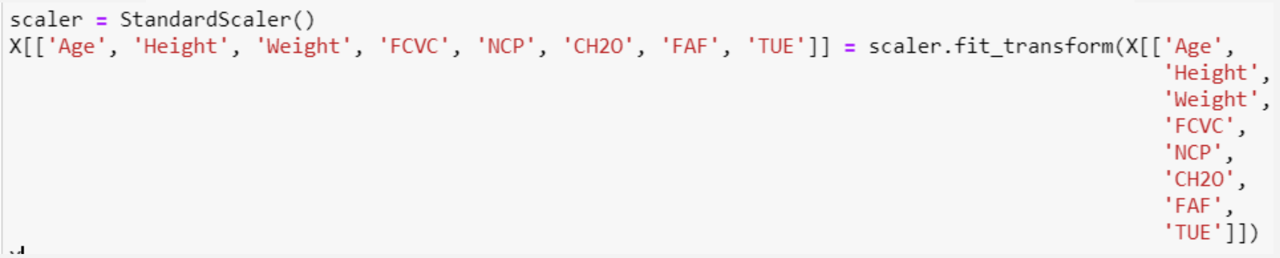
```
Xraw = obesity.drop('NObeyesdad', axis = 1)
yRaw = obesity['NObeyesdad']
```



```
encoder = LabelEncoder()
y = encoder.fit_transform(yRaw)
y = pd.DataFrame(y, columns=["NObeyesdad"])
```



```
X = pd.get_dummies(Xraw, columns = ['Gender',
                                    'family_history_with_overweight',
                                    'FAVC',
                                    'CAEC',
                                    'SMOKE',
                                    'SCC',
                                    'CALC',
                                    'MTRANS'
                                    ])
```



```
scaler = StandardScaler()
X[['Age', 'Height', 'Weight', 'FCVC', 'NCP', 'CH2O', 'FAF', 'TUE']] = scaler.fit_transform(X[['Age',
                                                                                              'Height',
                                                                                              'Weight',
                                                                                              'FCVC',
                                                                                              'NCP',
                                                                                              'CH2O',
                                                                                              'FAF',
                                                                                              'TUE']])
```

The description is done following the order of the images.

SPLITTING:

Splitting the dataset to create X (features) and y (target label)

ENCODING y:

Transforming the qualitative (text) data into numbers so that the ML algorithm can interpret the data.

ENCODING X:

Transforming the qualitative (text) data into numbers so that the ML algorithm can interpret the data. We are doing that on specific features as some are not qualitative.

NORMALIZING X:

Transforming the quantitative data so that they all have the same scale and rank as some algorithms tend to take into account the features' order.

LINEAR SVC

First algorithm of the three classification algorithm we are going to use.

FITTING THE STANDARD WAY (model.fit):

Train score: 78,4%

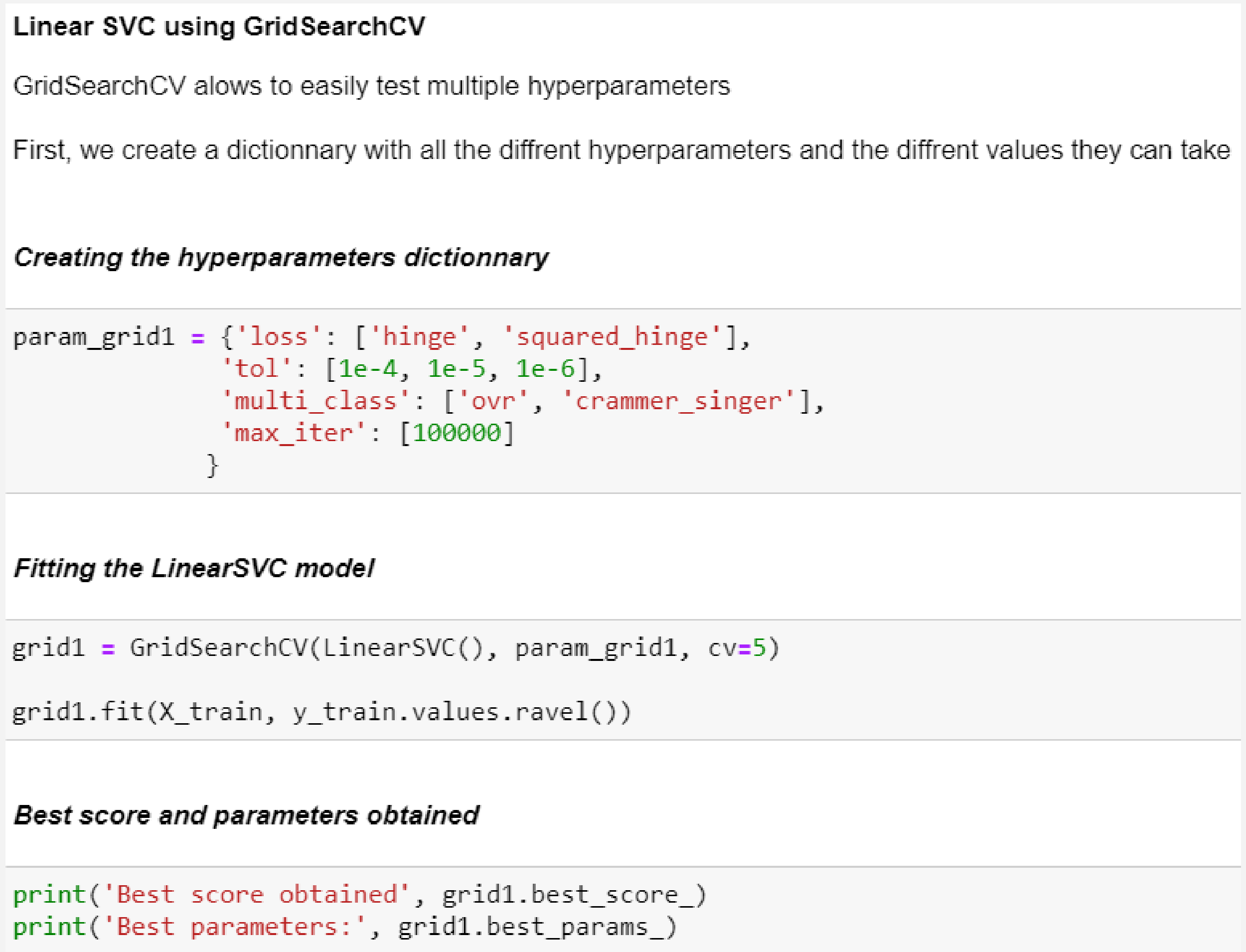
Test score:

FITTING USING GRIDSEARCHCV:

Refer to the screens on the right for the chosen parameters.

Best score obtained 0.9080422028913337

Best parameters: {'loss': 'hinge', 'max_iter': 100000, 'multi_class': 'crammer_singer', 'tol': 0.0001}



Linear SVC using GridSearchCV

GridSearchCV alows to easily test multiple hyperparameters

First, we create a dictionnary with all the diffrent hyperparameters and the diffrent values they can take

Creating the hyperparameters dictionnary

```
param_grid1 = {'loss': ['hinge', 'squared_hinge'],
               'tol': [1e-4, 1e-5, 1e-6],
               'multi_class': ['ovr', 'crammer_singer'],
               'max_iter': [100000]
              }
```

Fitting the LinearSVC model

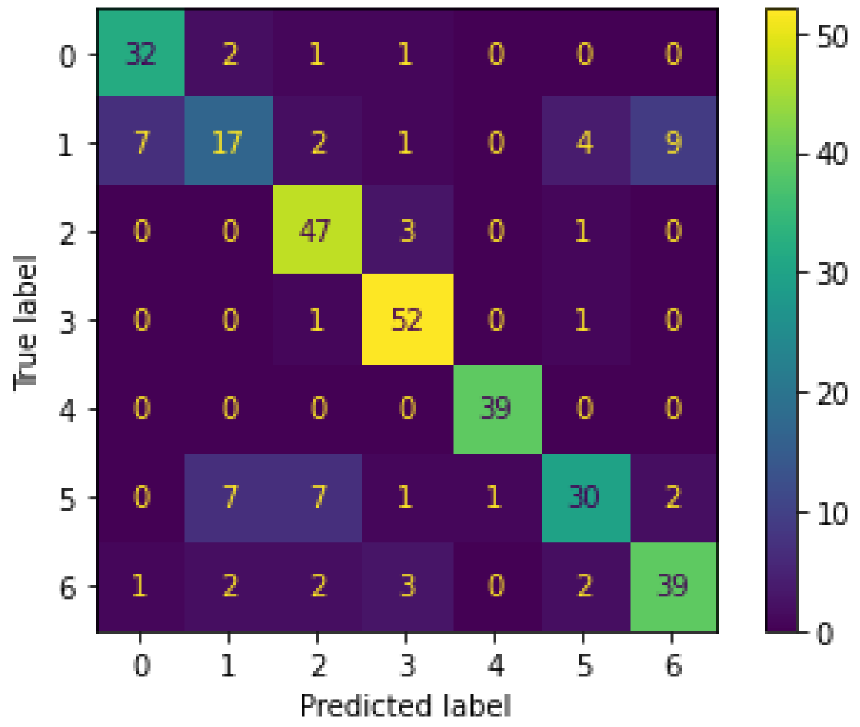
```
grid1 = GridSearchCV(LinearSVC(), param_grid1, cv=5)

grid1.fit(X_train, y_train.values.ravel())
```

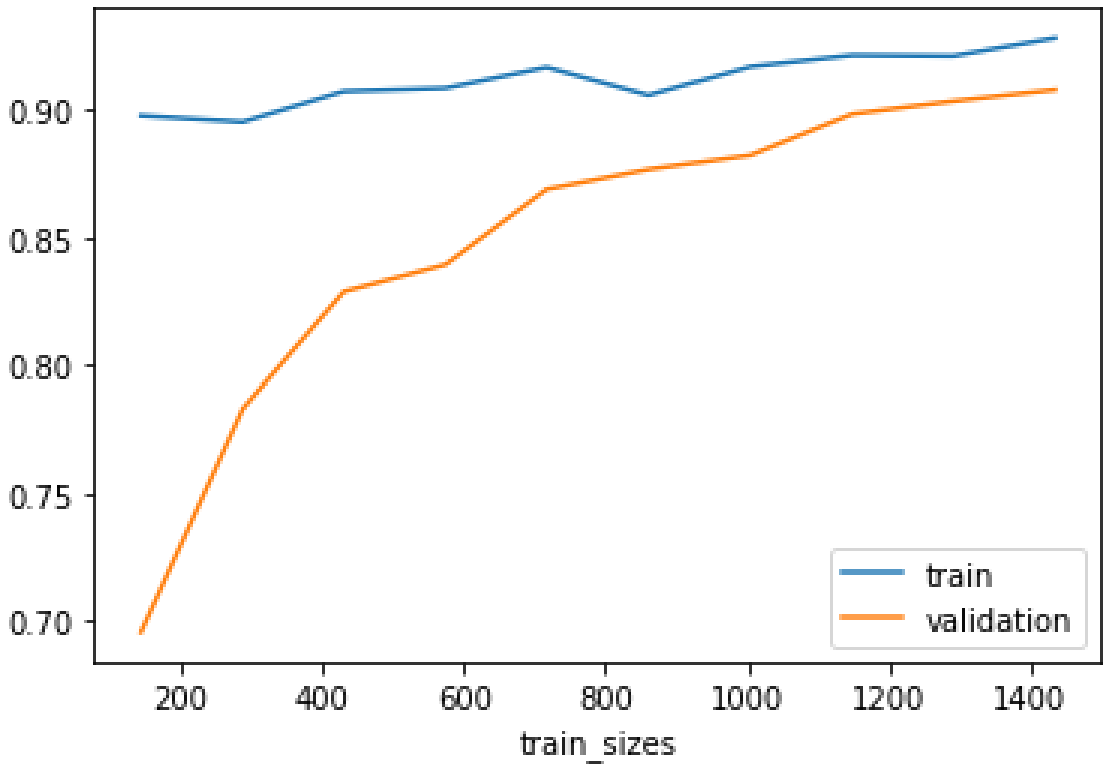
Best score and parameters obtained

```
print('Best score obtained', grid1.best_score_)
print('Best parameters:', grid1.best_params_)
```

LINEAR SVC



True label \ Predicted label	0	1	2	3	4	5	6
0	32	2	1	1	0	0	0
1	7	17	2	1	0	4	9
2	0	0	47	3	0	1	0
3	0	0	1	52	0	1	0
4	0	0	0	0	39	0	0
5	0	7	7	1	1	30	2
6	1	2	2	3	0	2	39



KNEIGHBORS CLASSIFIER

Second algorithm of the three classification algorithm we are going to use.

FITTING THE STANDARD WAY (model.fit):

Train score: 87,2%

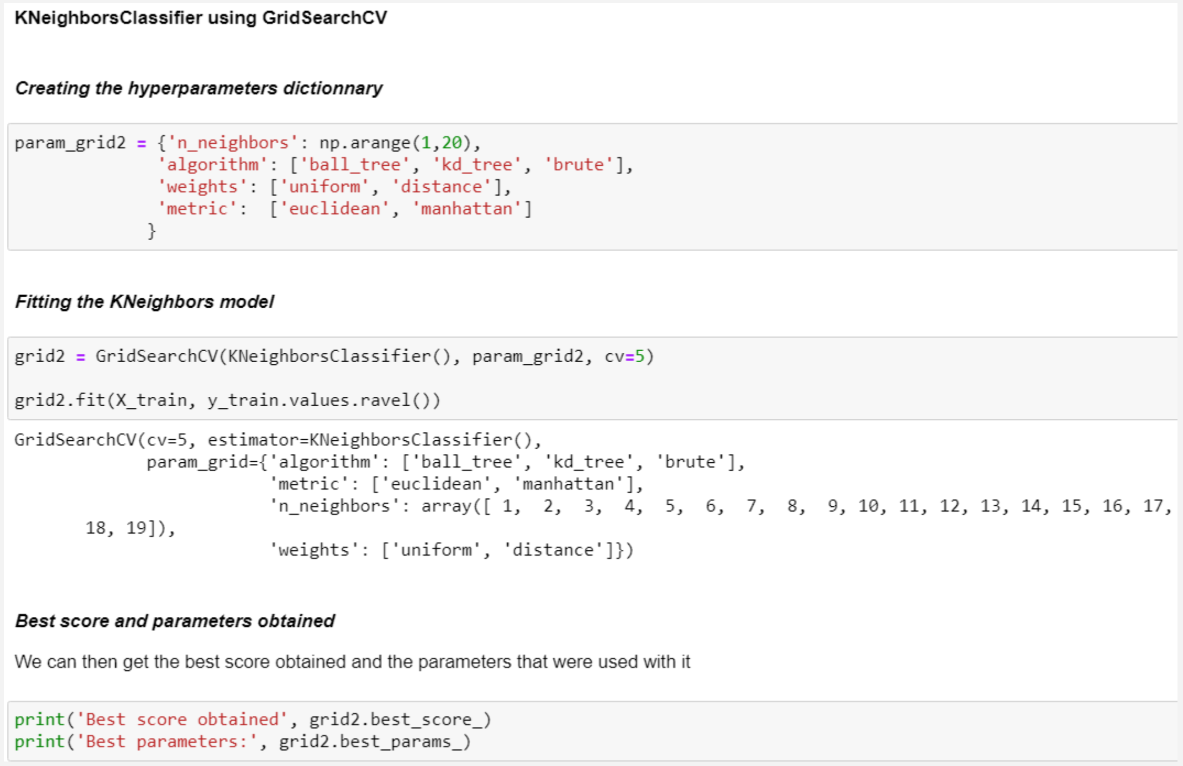
Test score: 77,2%

FITTING USING GRIDSEARCHCV:

Refer to the screens on the right for the chosen parameters.

Best score obtained 0.8533978618446646

Best parameters: {'algorithm': 'ball_tree', 'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'distance'}



KNeighborsClassifier using GridSearchCV

Creating the hyperparameters dictionnary

```
param_grid2 = {'n_neighbors': np.arange(1,20),
               'algorithm': ['ball_tree', 'kd_tree', 'brute'],
               'weights': ['uniform', 'distance'],
               'metric':  ['euclidean', 'manhattan']
              }
```

Fitting the KNeighbors model

```
grid2 = GridSearchCV(KNeighborsClassifier(), param_grid2, cv=5)

grid2.fit(X_train, y_train.values.ravel())
```

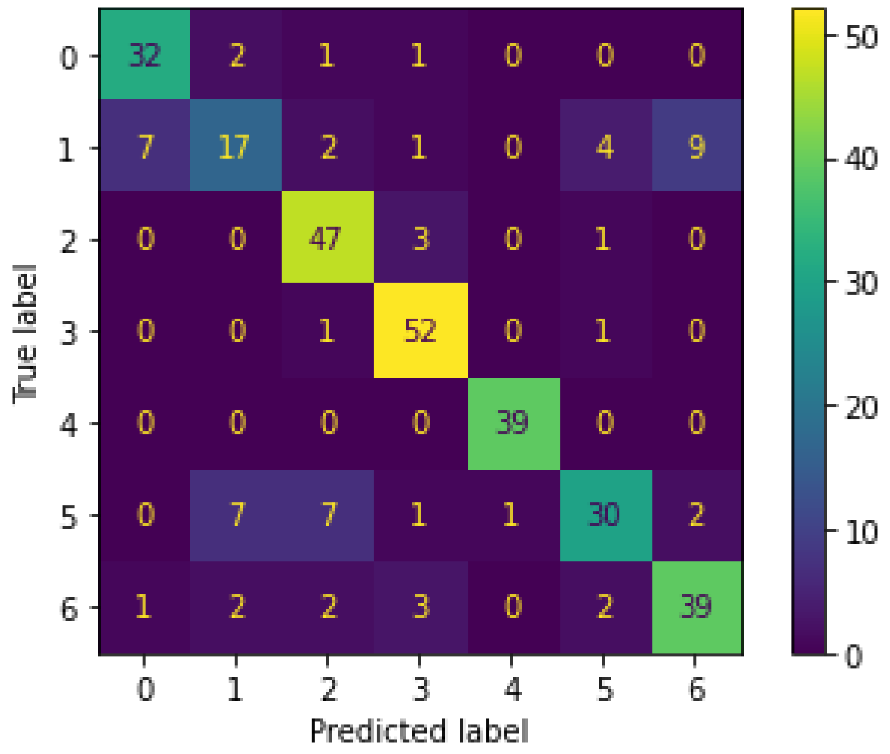
```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['ball_tree', 'kd_tree', 'brute'],
                         'metric': ['euclidean', 'manhattan'],
                         'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19]),
                         'weights': ['uniform', 'distance']})
```

Best score and parameters obtained

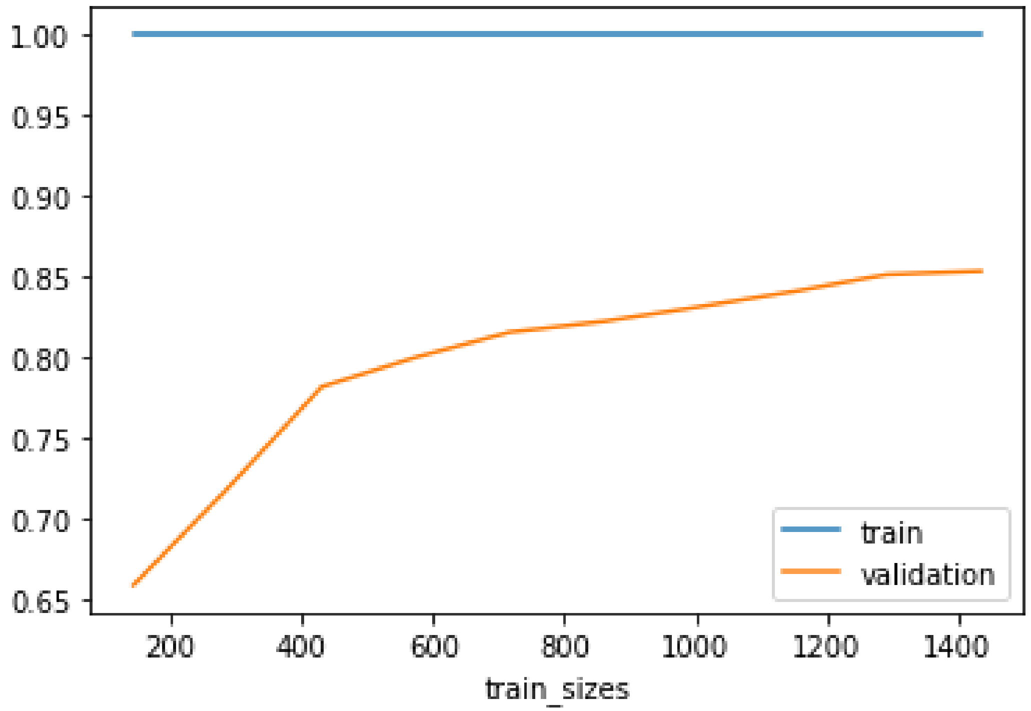
We can then get the best score obtained and the parameters that were used with it

```
print('Best score obtained', grid2.best_score_)
print('Best parameters:', grid2.best_params_)
```

KNEIGHBORS CLASSIFIER



	0	1	2	3	4	5	6
0	32	2	1	1	0	0	0
1	7	17	2	1	0	4	9
2	0	0	47	3	0	1	0
3	0	0	1	52	0	1	0
4	0	0	0	0	39	0	0
5	0	7	7	1	1	30	2
6	1	2	2	3	0	2	39



SVC

Third algorithm of the three classification algorithm we are going to use.

FITTING THE STANDARD WAY (model.fit):

Train score: 95,8%

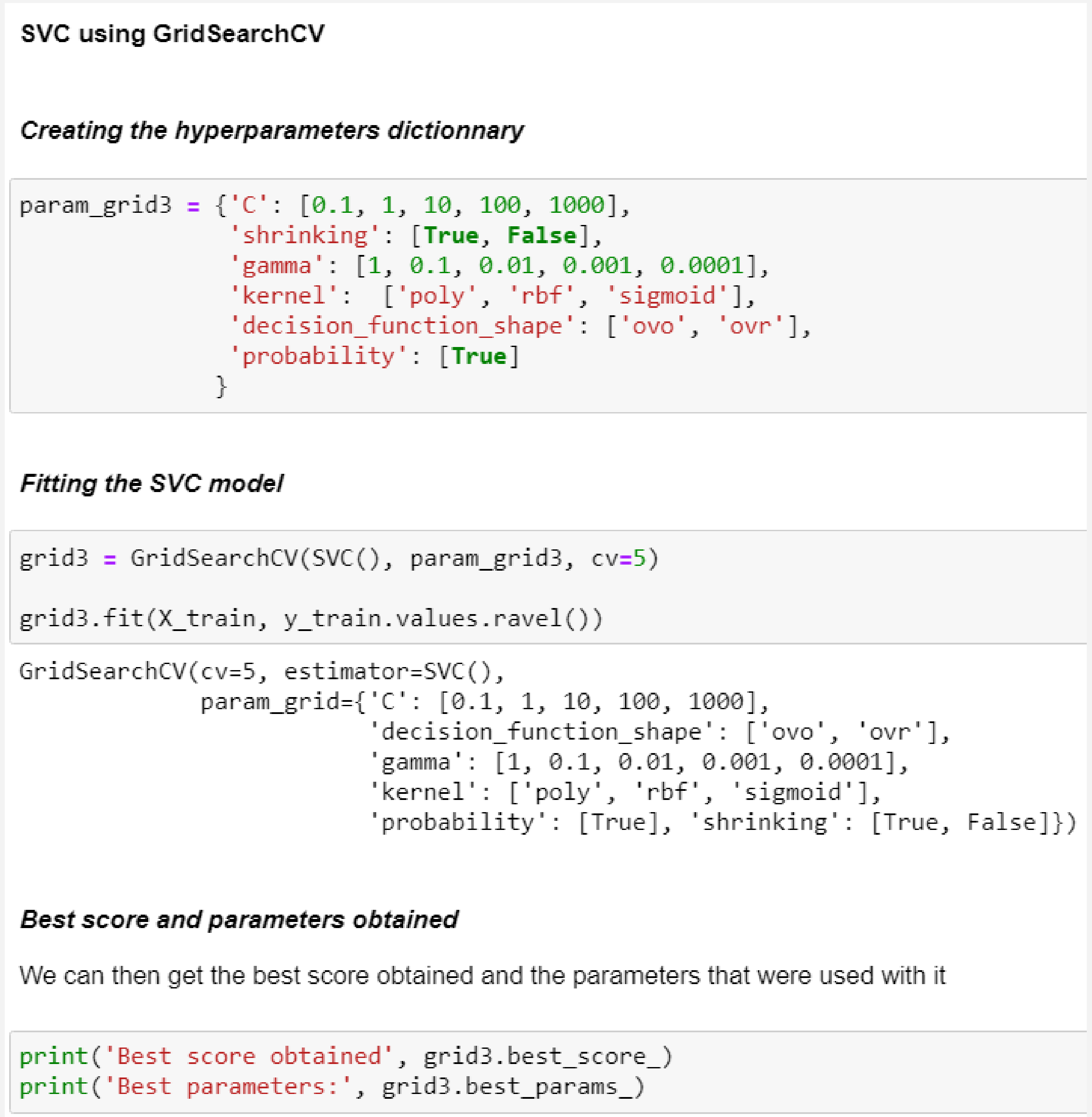
Test score:

FITTING USING GRIDSEARCHCV:

Refer to the screens on the right for the chosen parameters.

Best score obtained 0.9648807208104448

Best parameters: {'C': 1000, 'decision_function_shape': 'ovo', 'gamma': 0.001, 'kernel': 'rbf', 'probability': True, 'shrinking': True}



SVC using GridSearchCV

Creating the hyperparameters dictionnary

```
param_grid3 = {'C': [0.1, 1, 10, 100, 1000],
               'shrinking': [True, False],
               'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
               'kernel':  ['poly', 'rbf', 'sigmoid'],
               'decision_function_shape': ['ovo', 'ovr'],
               'probability': [True]
              }
```

Fitting the SVC model

```
grid3 = GridSearchCV(SVC(), param_grid3, cv=5)

grid3.fit(X_train, y_train.values.ravel())
```

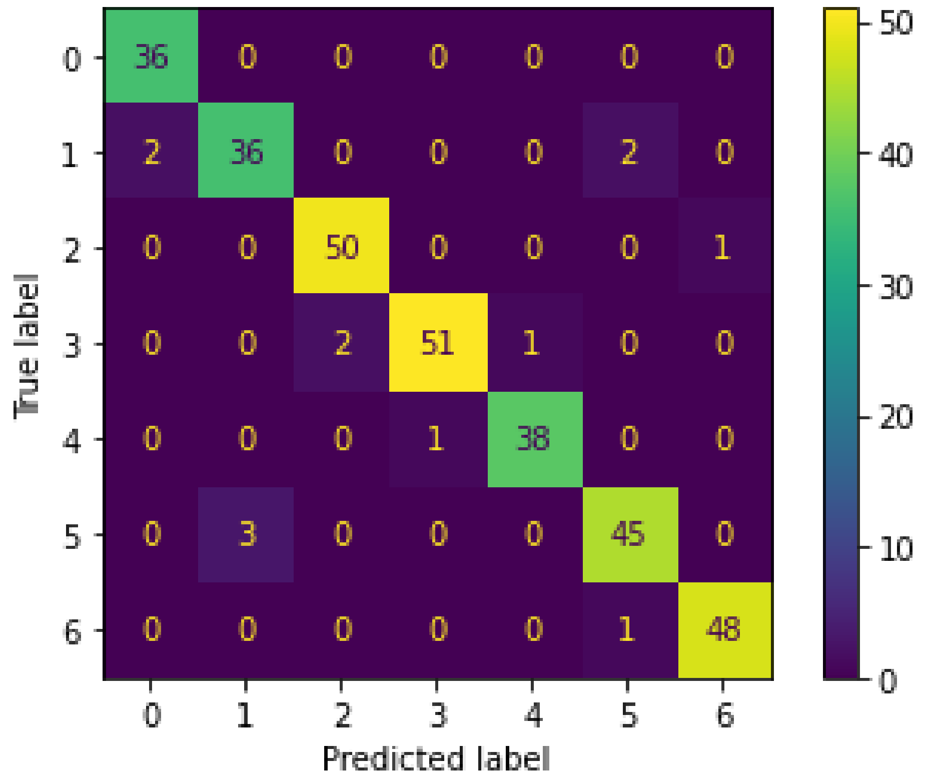
```
GridSearchCV(cv=5, estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100, 1000],
                         'decision_function_shape': ['ovo', 'ovr'],
                         'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
                         'kernel': ['poly', 'rbf', 'sigmoid'],
                         'probability': [True], 'shrinking': [True, False]})
```

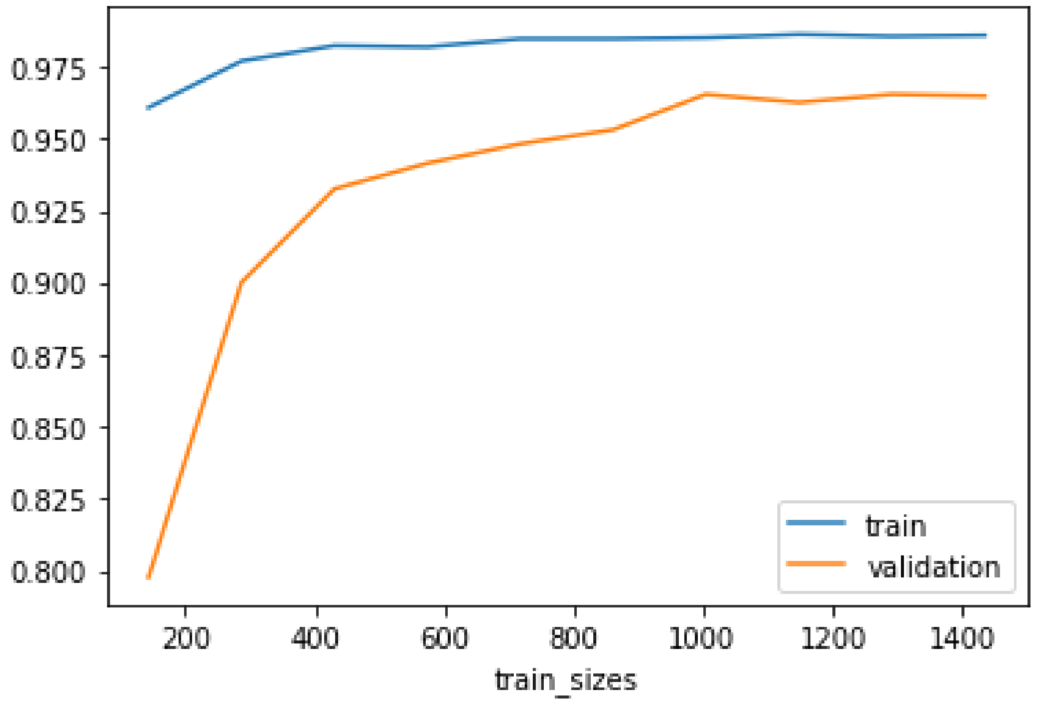
Best score and parameters obtained

We can then get the best score obtained and the parameters that were used with it

```
print('Best score obtained', grid3.best_score_)
print('Best parameters:', grid3.best_params_)
```

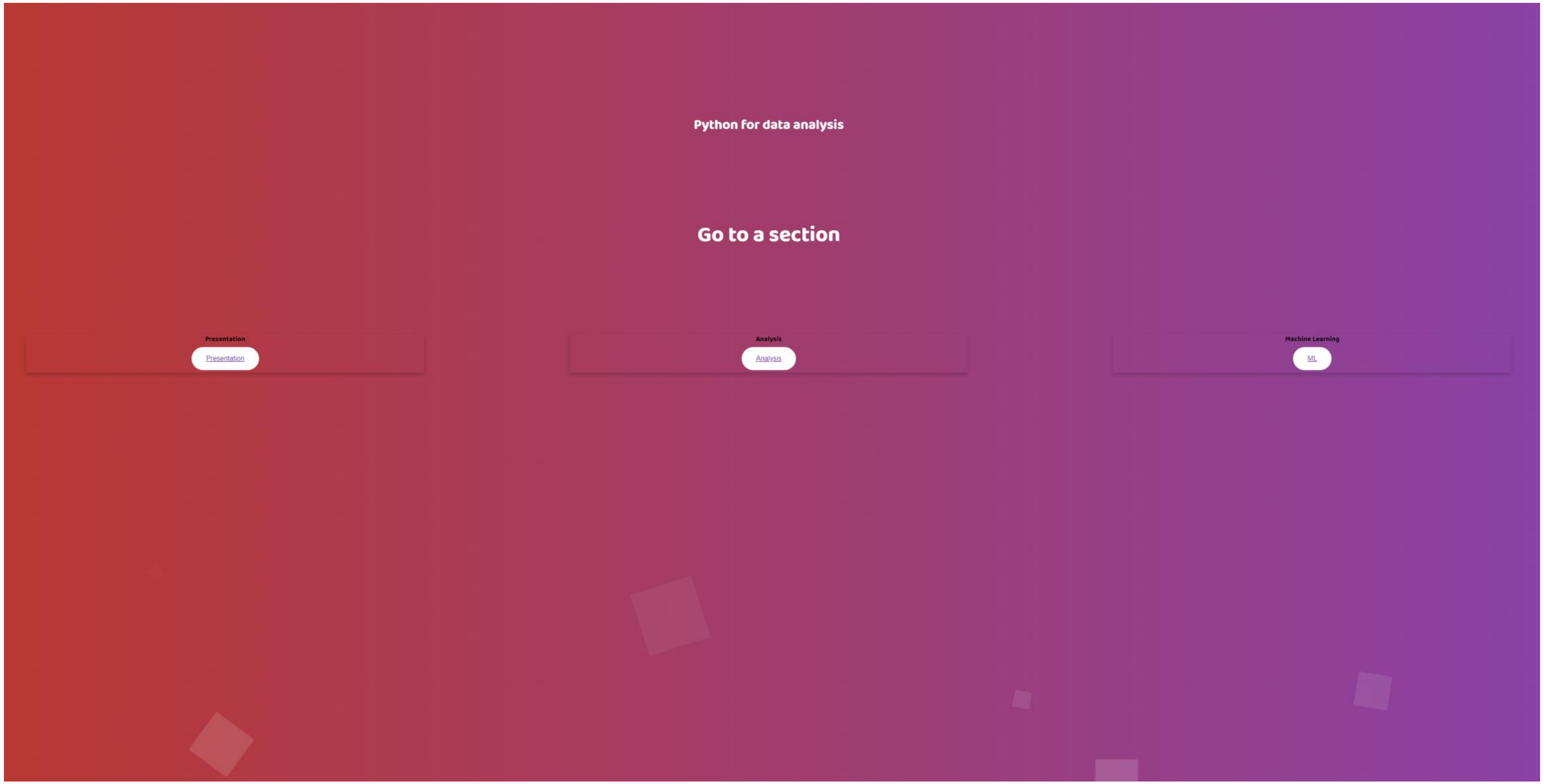
SVC


True label
Predicted label
0
1
2
3
4
5
6
36 0 0 0 0 0 0
2 36 0 0 0 2 0
0 0 50 0 0 0 1
0 0 2 51 1 0 0
0 0 0 1 38 0 0
0 3 0 0 0 45 0
0 0 0 0 0 1 48
0
10
20
30
40
50


0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
200
400
600
800
1000
1200
1400
train_sizes
train
validation

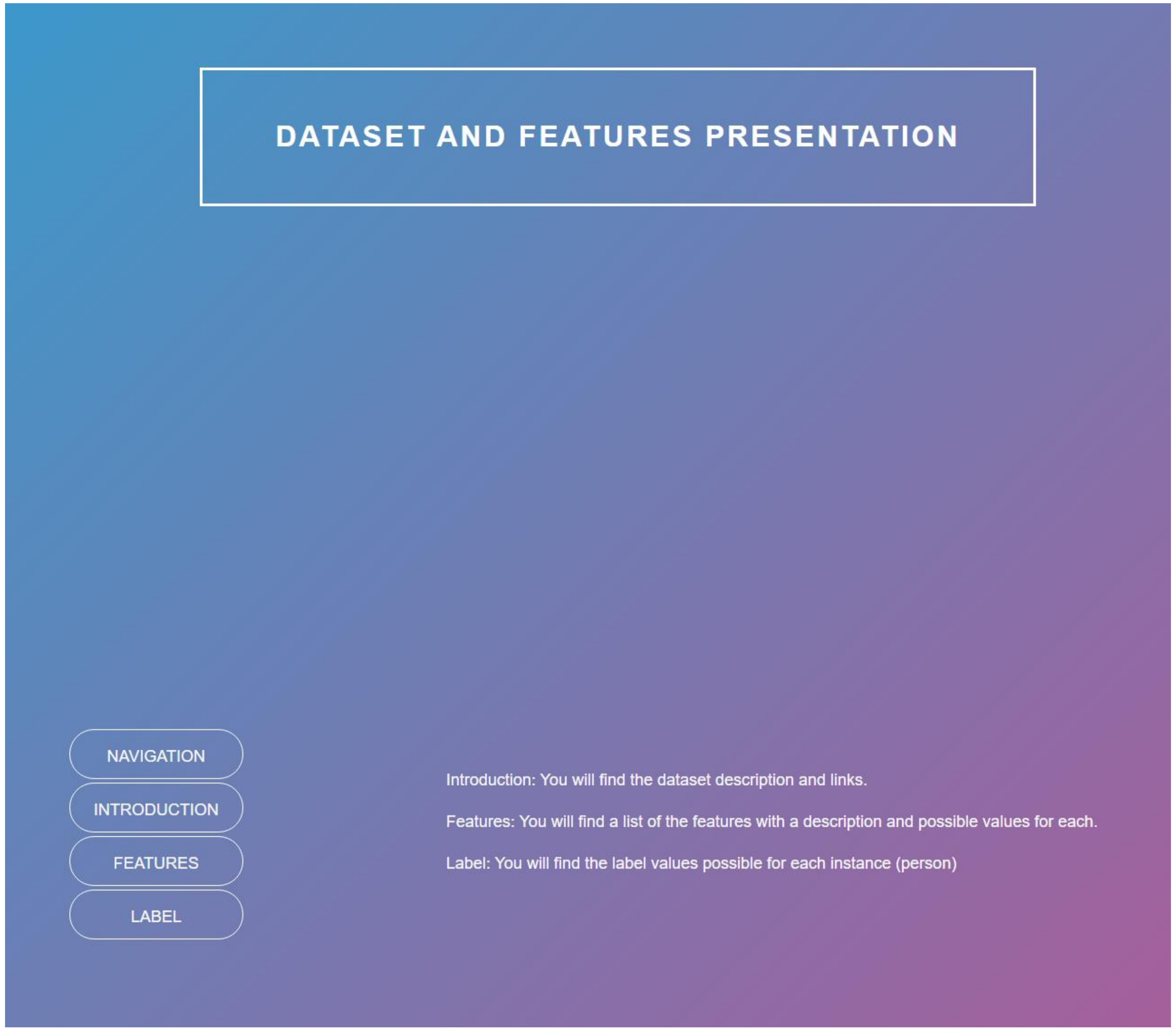
API - DJANGO

Lien video API : https://drive.google.com/drive/folders/116FZo2X-KakjZoefeBZLfeswrGCsDZ0C?usp=sharing


Python for data analysis

Go to a section

PRESENTATION

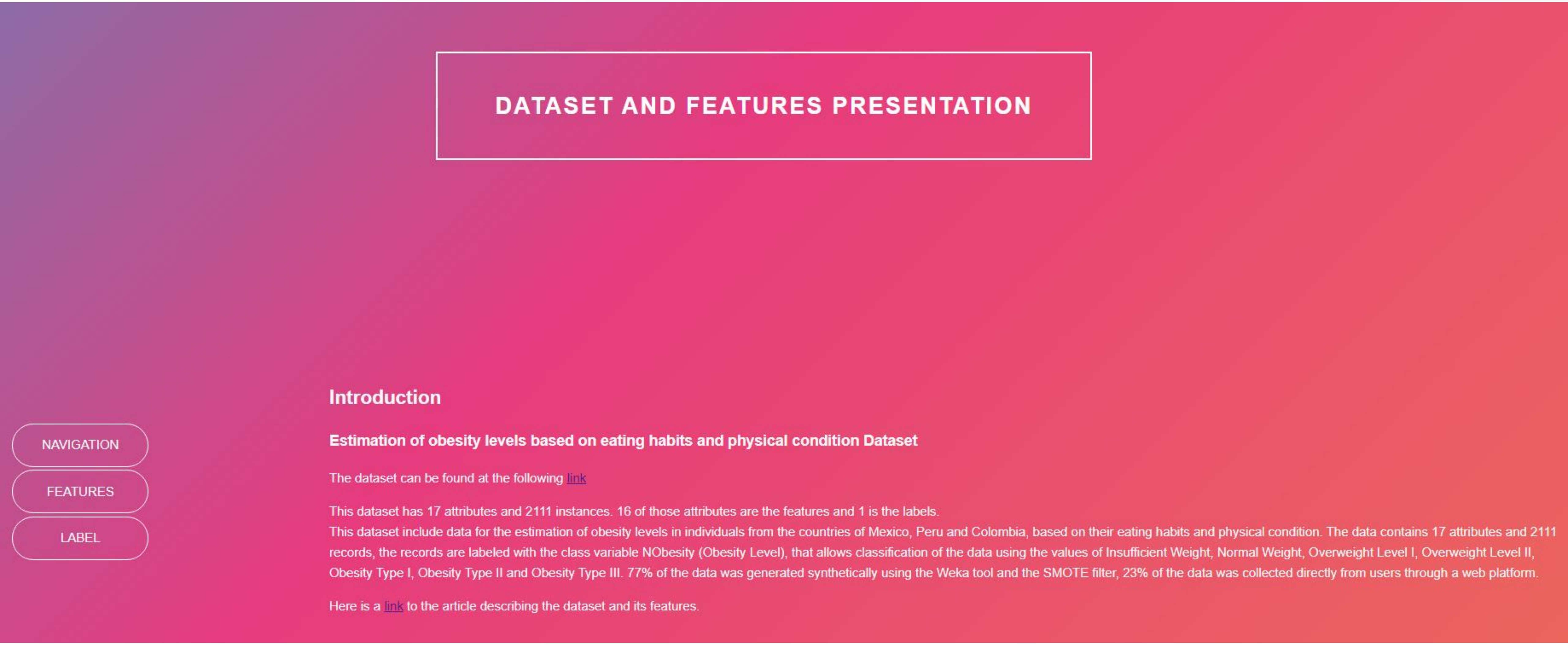


Introduction: You will find the dataset description and links.

Features: You will find a list of the features with a description and possible values for each.

Label: You will find the label values possible for each instance (person)

PRESENTATION



DATASET AND FEATURES PRESENTATION

Introduction

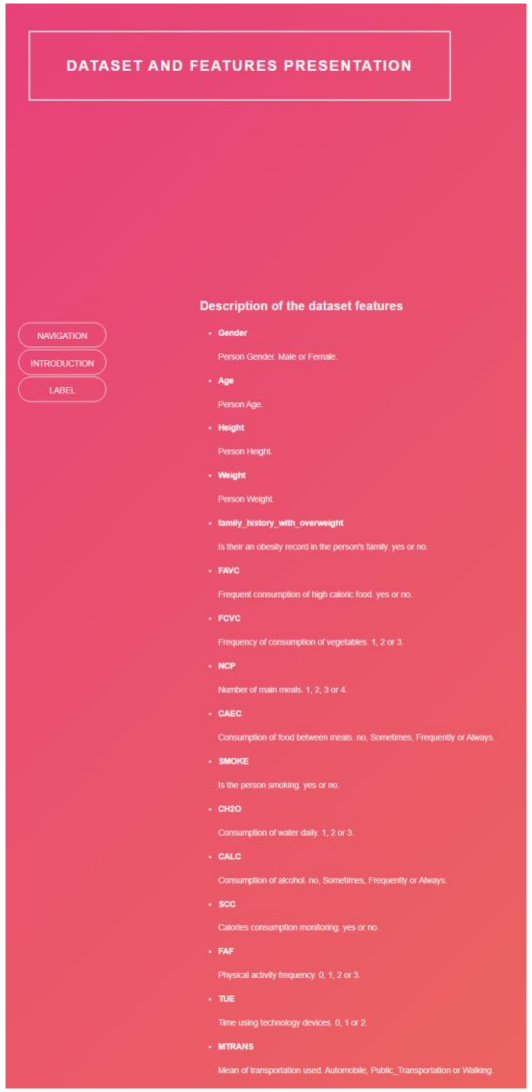
Estimation of obesity levels based on eating habits and physical condition Dataset

The dataset can be found at the following link

This dataset has 17 attributes and 2111 instances. 16 of those attributes are the features and 1 is the labels.
This dataset include data for the estimation of obesity levels in individuals from the countries of Mexico, Peru and Colombia, based on their eating habits and physical condition. 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. 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.

Here is a link to the article describing the dataset and its features.

PRESENTATION



Description of the dataset features

- **Gender**

  Person Gender. Male or Female.
- **Age**

  Person Age.
- **Height**

  Person Height.
- **Weight**

  Person Weight.
- **family_history_with_overweight**

  Is their an obesity record in the person's family. yes or no.
- **FAVC**

  Frequent consumption of high caloric food. yes or no.
- **FCVC**

  Frequency of consumption of vegetables. 1, 2 or 3.
- **NCP**

  Number of main meals. 1, 2, 3 or 4.
- **CAEC**

  Consumption of food between meals. no, Sometimes, Frequently or Always.
- **SMOKE**

  Is the person smoking. yes or no.
- **CH2O**

  Consumption of water daily. 1, 2 or 3.
- **CALC**

  Consumption of alcohol. no, Sometimes, Frequently or Always.
- **SCC**

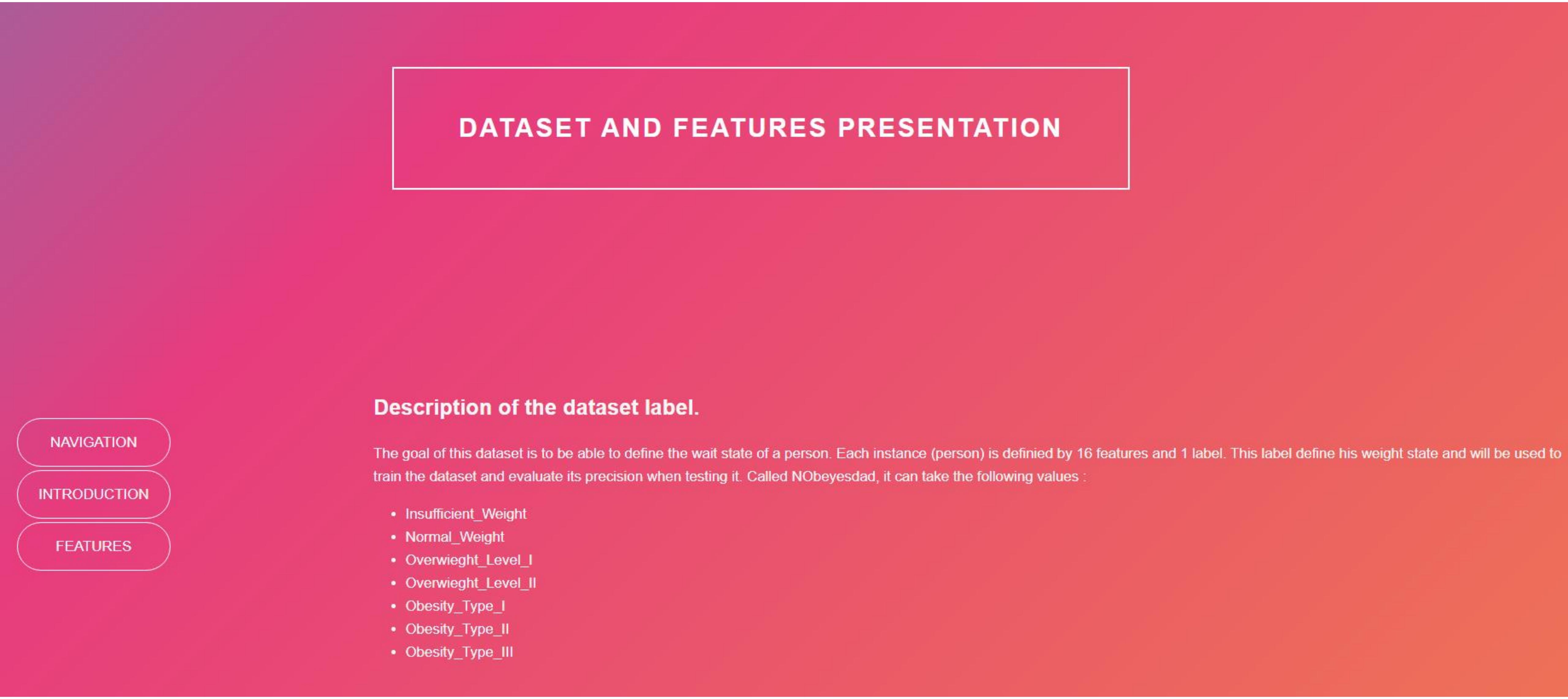
  Calories consumption monitoring. yes or no.
- **FAF**

  Physical activity frequency. 0, 1, 2 or 3.
- **TUE**

  Time using technology devices. 0, 1 or 2.
- **MTRANS**

  Mean of transportation used. Automobile, Public_Transportation or Walking.

PRESENTATION



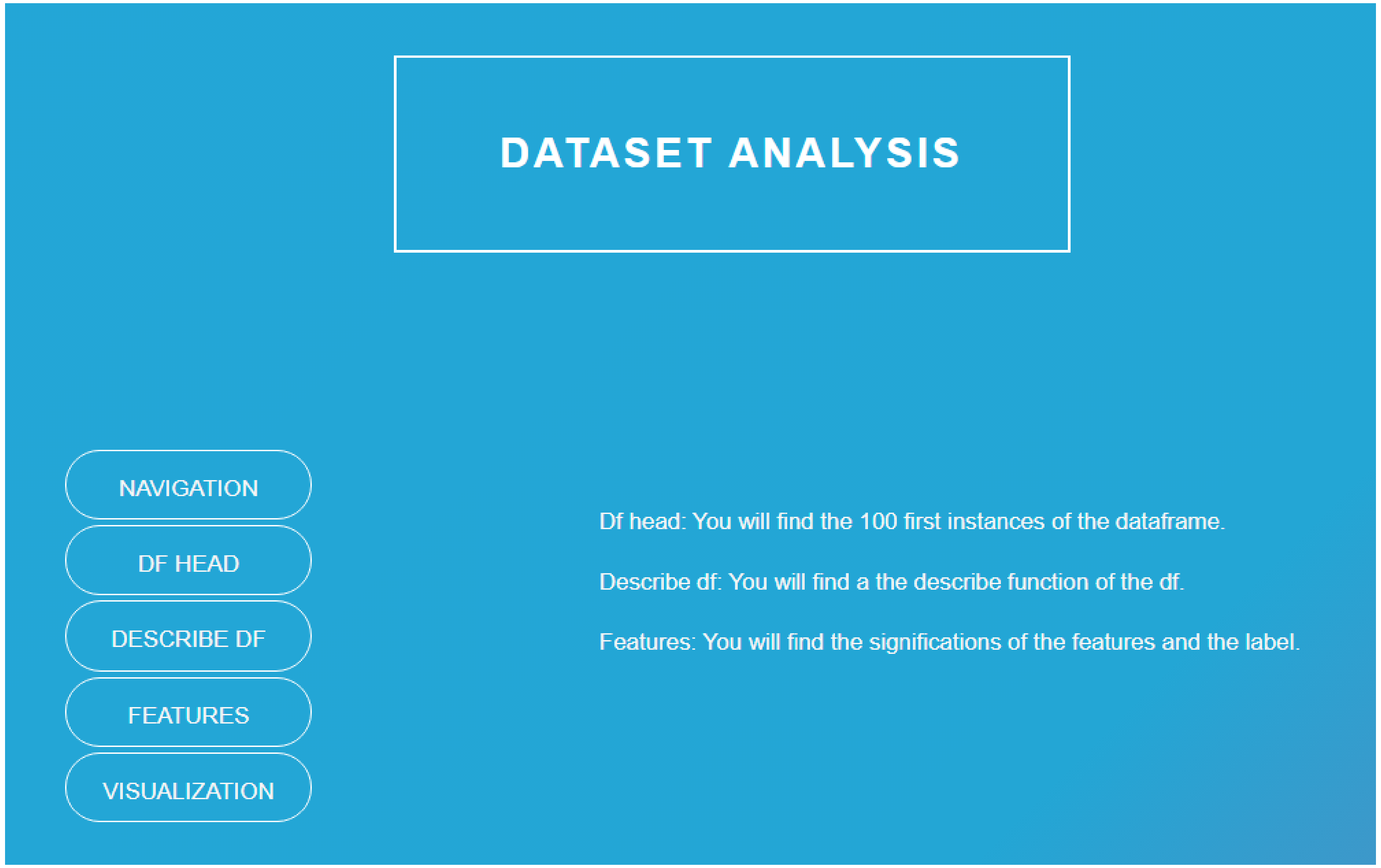
DATASET AND FEATURES PRESENTATION

Description of the dataset label.

The goal of this dataset is to be able to define the wait state of a person. Each instance (person) is definied by 16 features and 1 label. This label define his weight state and will be used to train the dataset and evaluate its precision when testing it. Called NObeyesdad, it can take the following values :

- Insufficient_Weight
- Normal_Weight
- Overwieght_Level_I
- Overwieght_Level_II
- Obesity_Type_I
- Obesity_Type_II
- Obesity_Type_III

ANALYSIS



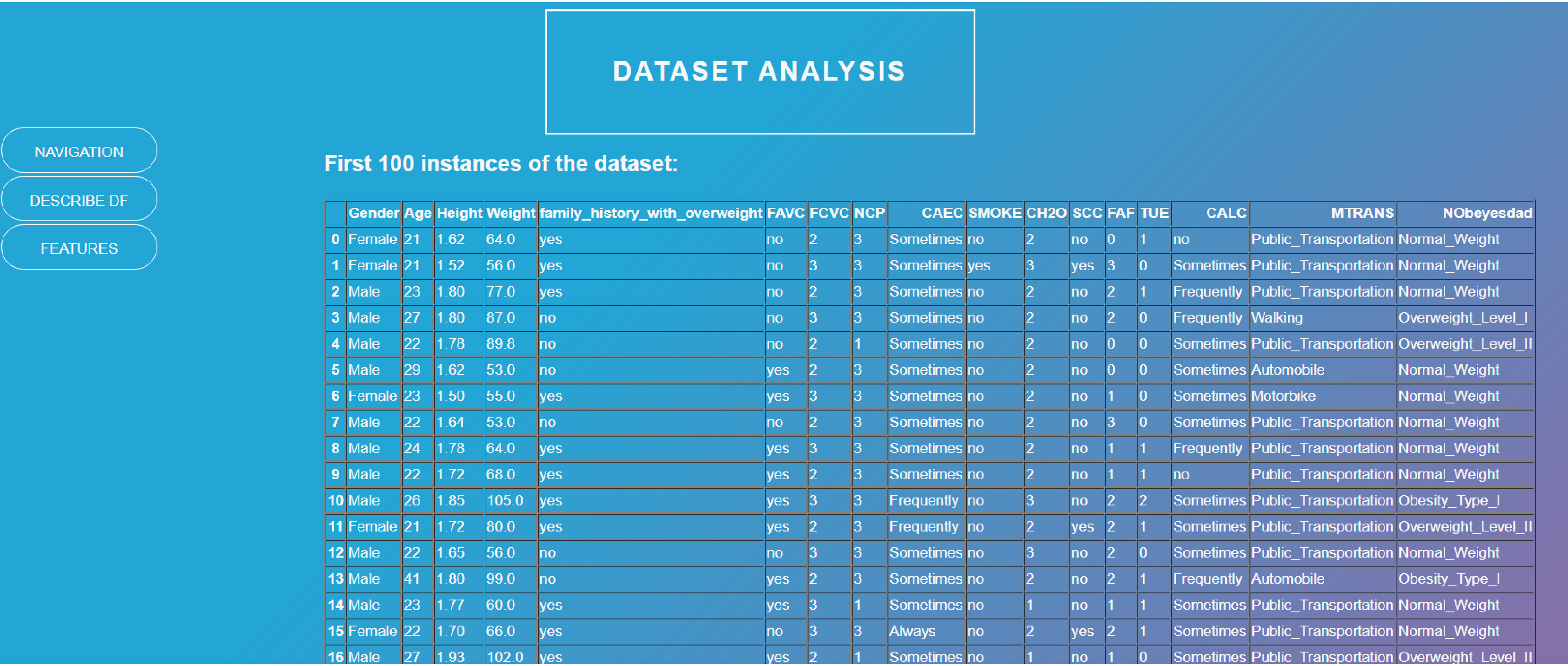
DATASET ANALYSIS

Df head: You will find the 100 first instances of the dataframe.

Describe df: You will find a the describe function of the df.

Features: You will find the significations of the features and the label.

ANALYSIS

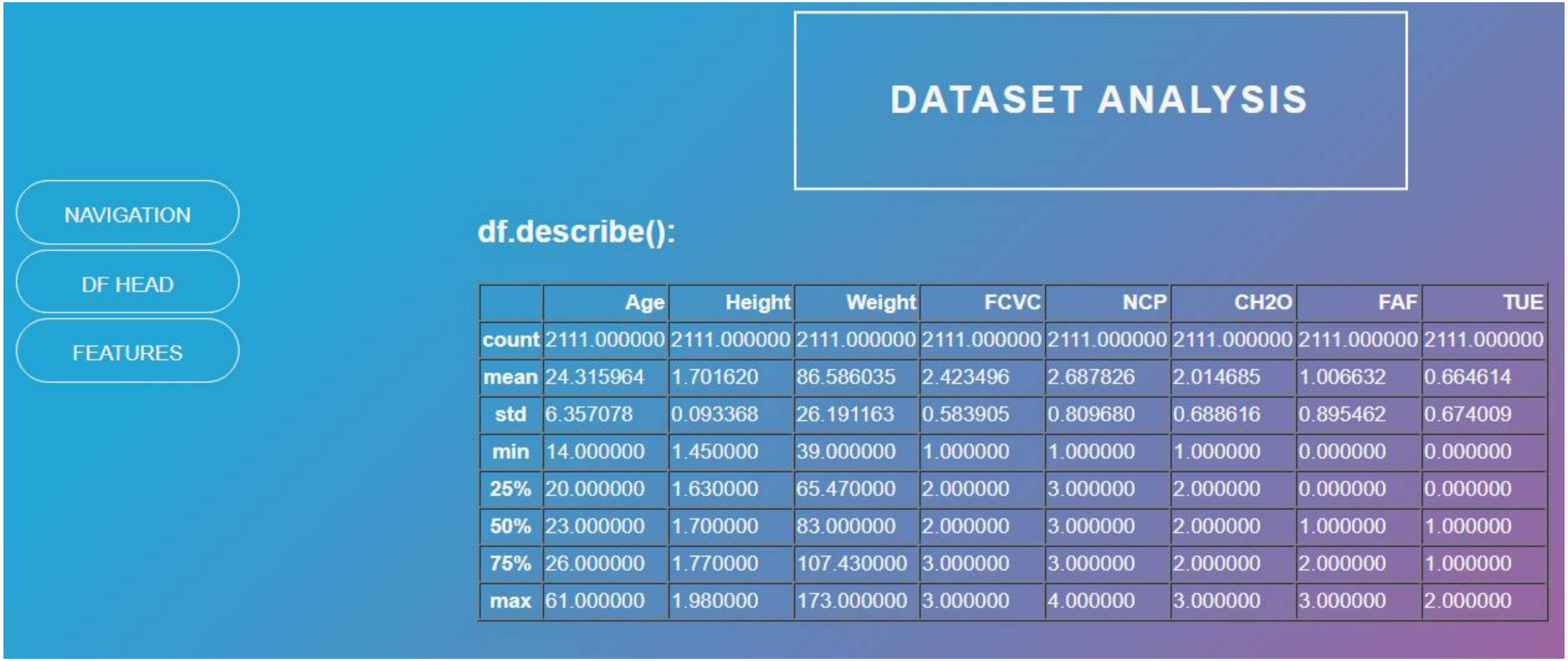


DATASET ANALYSIS

First 100 instances of the dataset:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	NObeyesdad
0	Female	21	1.62	64.0	yes	no	2	3	Sometimes	no	2	no	0	1	no	Public_Transportation	Normal_Weight
1	Female	21	1.52	56.0	yes	no	3	3	Sometimes	yes	3	yes	3	0	Sometimes	Public_Transportation	Normal_Weight
2	Male	23	1.80	77.0	yes	no	2	3	Sometimes	no	2	no	2	1	Frequently	Public_Transportation	Normal_Weight
3	Male	27	1.80	87.0	no	no	3	3	Sometimes	no	2	no	2	0	Frequently	Walking	Overweight_Level_I
4	Male	22	1.78	89.8	no	no	2	1	Sometimes	no	2	no	0	0	Sometimes	Public_Transportation	Overweight_Level_II
5	Male	29	1.62	53.0	no	yes	2	3	Sometimes	no	2	no	0	0	Sometimes	Automobile	Normal_Weight
6	Female	23	1.50	55.0	yes	yes	3	3	Sometimes	no	2	no	1	0	Sometimes	Motorbike	Normal_Weight
7	Male	22	1.64	53.0	no	no	2	3	Sometimes	no	2	no	3	0	Sometimes	Public_Transportation	Normal_Weight
8	Male	24	1.78	64.0	yes	yes	3	3	Sometimes	no	2	no	1	1	Frequently	Public_Transportation	Normal_Weight
9	Male	22	1.72	68.0	yes	yes	2	3	Sometimes	no	2	no	1	1	no	Public_Transportation	Normal_Weight
10	Male	26	1.85	105.0	yes	yes	3	3	Frequently	no	3	no	2	2	Sometimes	Public_Transportation	Obesity_Type_I
11	Female	21	1.72	80.0	yes	yes	2	3	Frequently	no	2	yes	2	1	Sometimes	Public_Transportation	Overweight_Level_II
12	Male	22	1.65	56.0	no	no	3	3	Sometimes	no	3	no	2	0	Sometimes	Public_Transportation	Normal_Weight
13	Male	41	1.80	99.0	no	yes	2	3	Sometimes	no	2	no	2	1	Frequently	Automobile	Obesity_Type_I
14	Male	23	1.77	60.0	yes	yes	3	1	Sometimes	no	1	no	1	1	Sometimes	Public_Transportation	Normal_Weight
15	Female	22	1.70	66.0	yes	no	3	3	Always	no	2	yes	2	1	Sometimes	Public_Transportation	Normal_Weight
16	Male	27	1.93	102.0	yes	yes	2	1	Sometimes	no	1	no	1	0	Sometimes	Public_Transportation	Overweight_Level_II

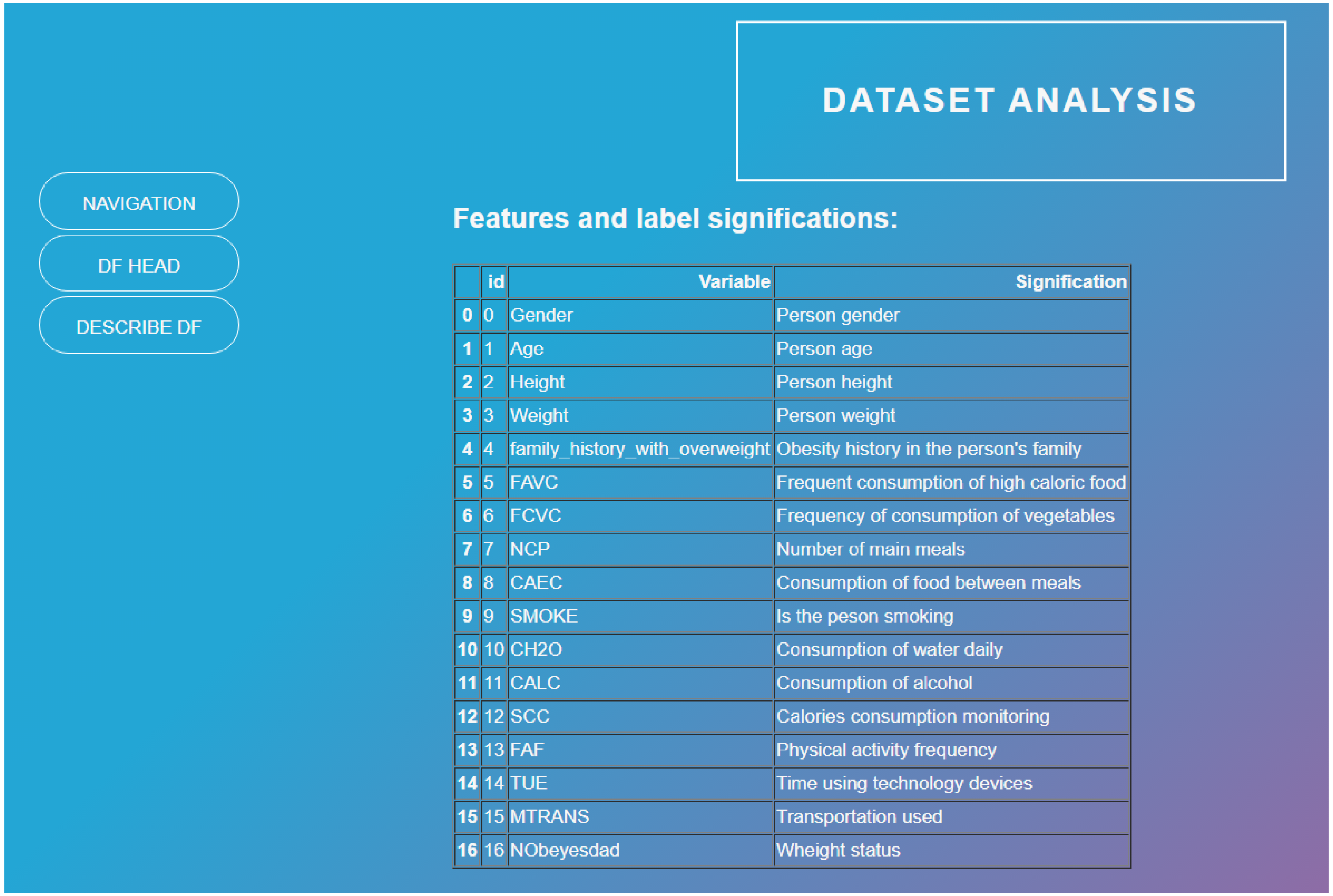
ANALYSIS



df.describe():

	Age	Height	Weight	FCVC	NCP	CH2O	FAF	TUE
count	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000
mean	24.315964	1.701620	86.586035	2.423496	2.687826	2.014685	1.006632	0.664614
std	6.357078	0.093368	26.191163	0.583905	0.809680	0.688616	0.895462	0.674009
min	14.000000	1.450000	39.000000	1.000000	1.000000	1.000000	0.000000	0.000000
25%	20.000000	1.630000	65.470000	2.000000	3.000000	2.000000	0.000000	0.000000
50%	23.000000	1.700000	83.000000	2.000000	3.000000	2.000000	1.000000	1.000000
75%	26.000000	1.770000	107.430000	3.000000	3.000000	2.000000	2.000000	1.000000
max	61.000000	1.980000	173.000000	3.000000	4.000000	3.000000	3.000000	2.000000

ANALYSIS

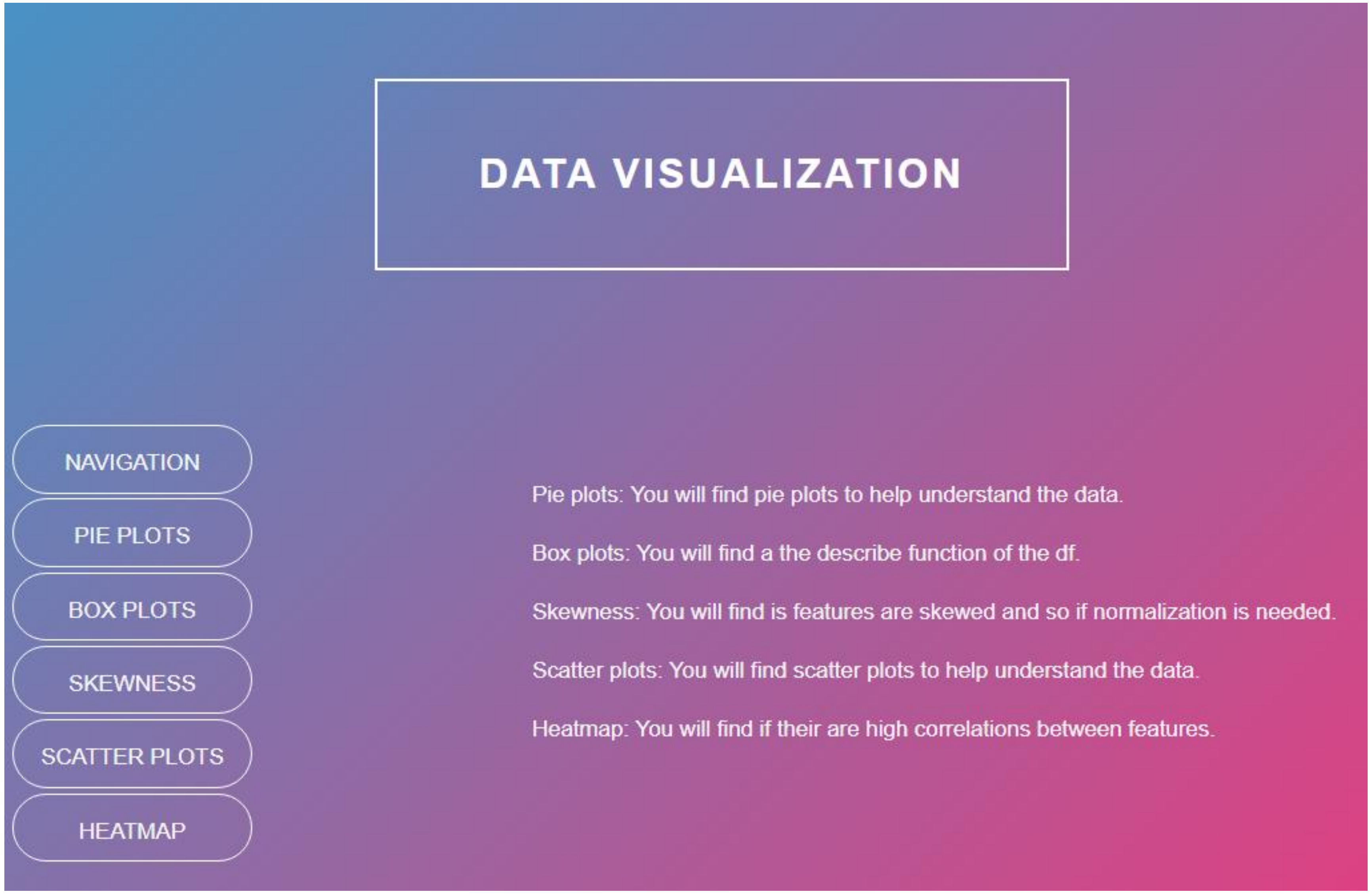


DATASET ANALYSIS

Features and label significations:

	id	Variable	Signification
0	0	Gender	Person gender
1	1	Age	Person age
2	2	Height	Person height
3	3	Weight	Person weight
4	4	family_history_with_overweight	Obesity history in the person's family
5	5	FAVC	Frequent consumption of high caloric food
6	6	FCVC	Frequency of consumption of vegetables
7	7	NCP	Number of main meals
8	8	CAEC	Consumption of food between meals
9	9	SMOKE	Is the peson smoking
10	10	CH2O	Consumption of water daily
11	11	CALC	Consumption of alcohol
12	12	SCC	Calories consumption monitoring
13	13	FAF	Physical activity frequency
14	14	TUE	Time using technology devices
15	15	MTRANS	Transportation used
16	16	NObeyesdad	Wheight status

ANALYSIS



Pie plots: You will find pie plots to help understand the data.

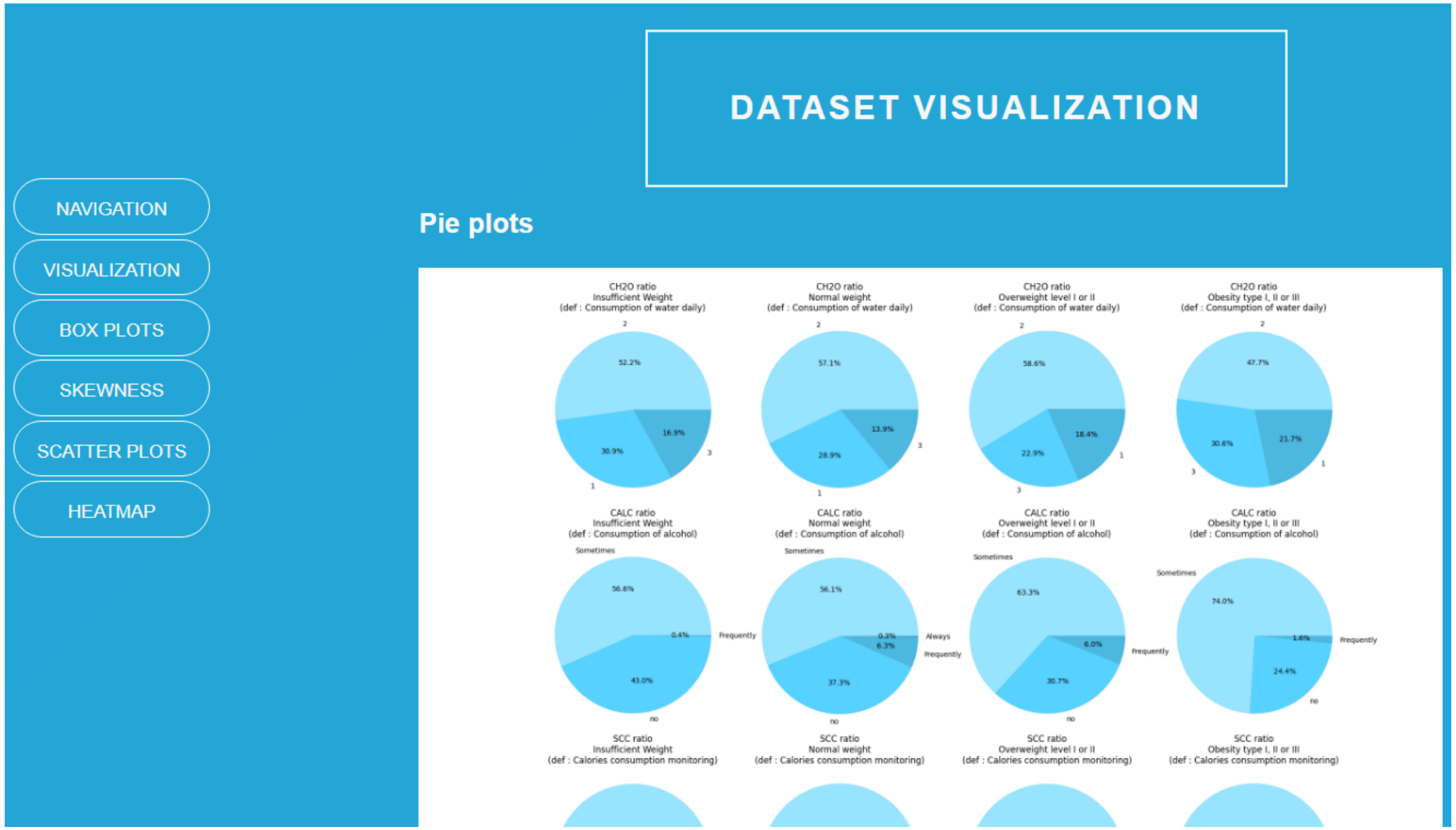
Box plots: You will find a the describe function of the df.

Skewness: You will find is features are skewed and so if normalization is needed.

Scatter plots: You will find scatter plots to help understand the data.

Heatmap: You will find if their are high correlations between features.

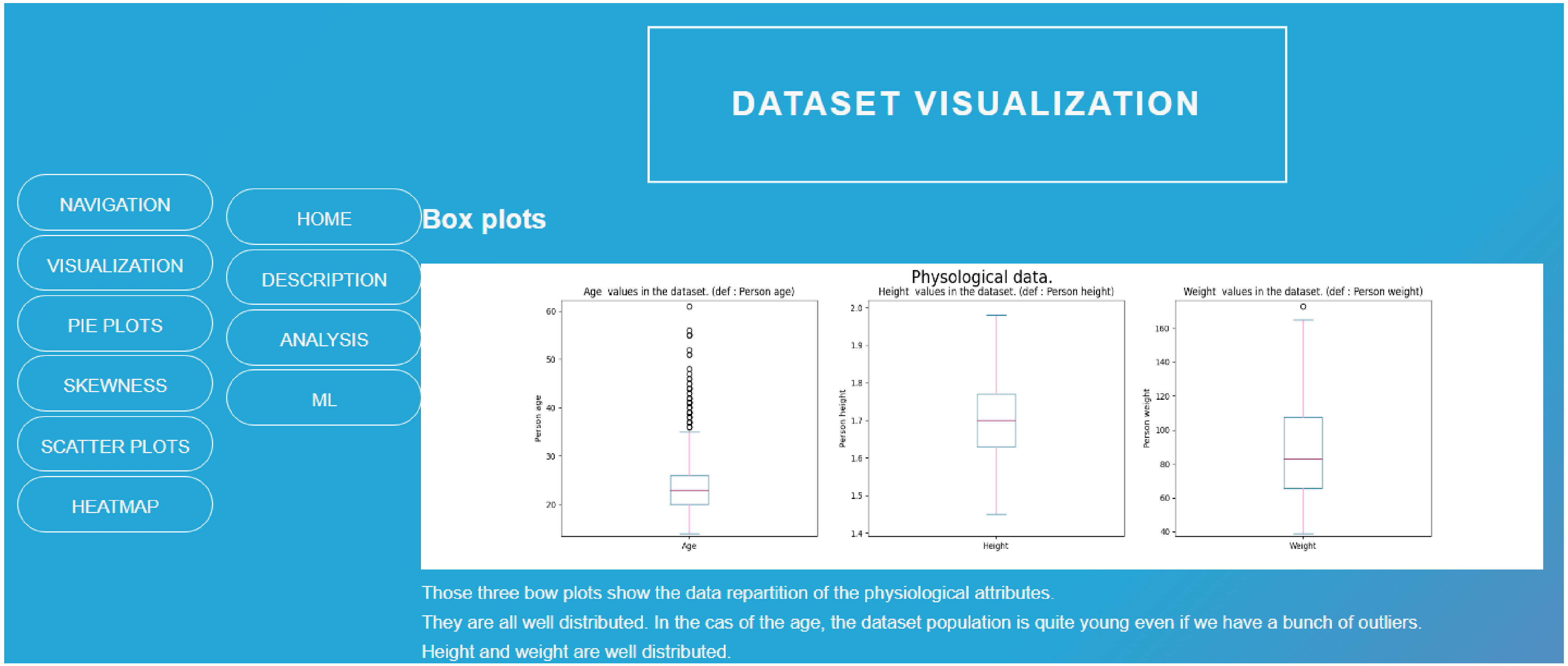
ANALYSIS


NAVIGATION
VISUALIZATION
BOX PLOTS
SKEWNESS
SCATTER PLOTS
HEATMAP
CH2O ratio Insufficient Weight (def : Consumption of water daily)
2
52.2%
30.9%
16.9%
1
3
CH2O ratio Normal weight (def : Consumption of water daily)
57.1%
28.9%
13.9%
CH2O ratio Overweight level I or II (def : Consumption of water daily)
58.6%
22.9%
18.4%
CH2O ratio Obesity type I, II or III (def : Consumption of water daily)
47.7%
30.6%
21.7%
CALC ratio Insufficient Weight (def : Consumption of alcohol)
Sometimes
56.6%
43.0%
0.4%
Frequently
no
CALC ratio Normal weight (def : Consumption of alcohol)
56.1%
37.3%
0.3%
6.3%
Always
CALC ratio Overweight level I or II (def : Consumption of alcohol)
63.3%
30.7%
6.0%
CALC ratio Obesity type I, II or III (def : Consumption of alcohol)
74.0%
24.4%
1.6%
SCC ratio Insufficient Weight (def : Calories consumption monitoring)
SCC ratio Normal weight (def : Calories consumption monitoring)
SCC ratio Overweight level I or II (def : Calories consumption monitoring)
SCC ratio Obesity type I, II or III (def : Calories consumption monitoring)

DATASET VISUALIZATION

Pie plots

ANALYSIS



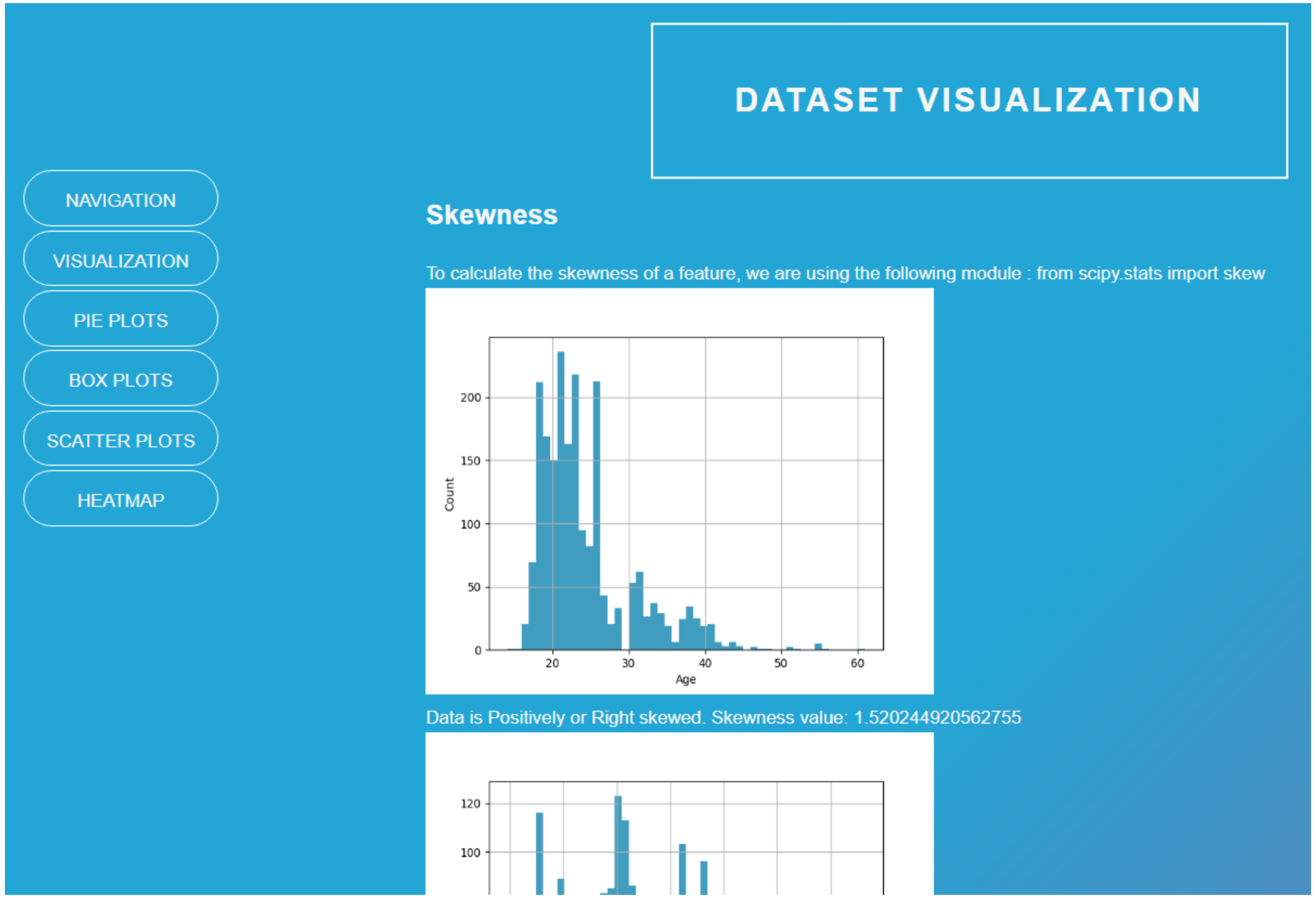
Box plots

Those three bow plots show the data repartition of the physiological attributes.

They are all well distributed. In the cas of the age, the dataset population is quite young even if we have a bunch of outliers.

Height and weight are well distributed.

ANALYSIS



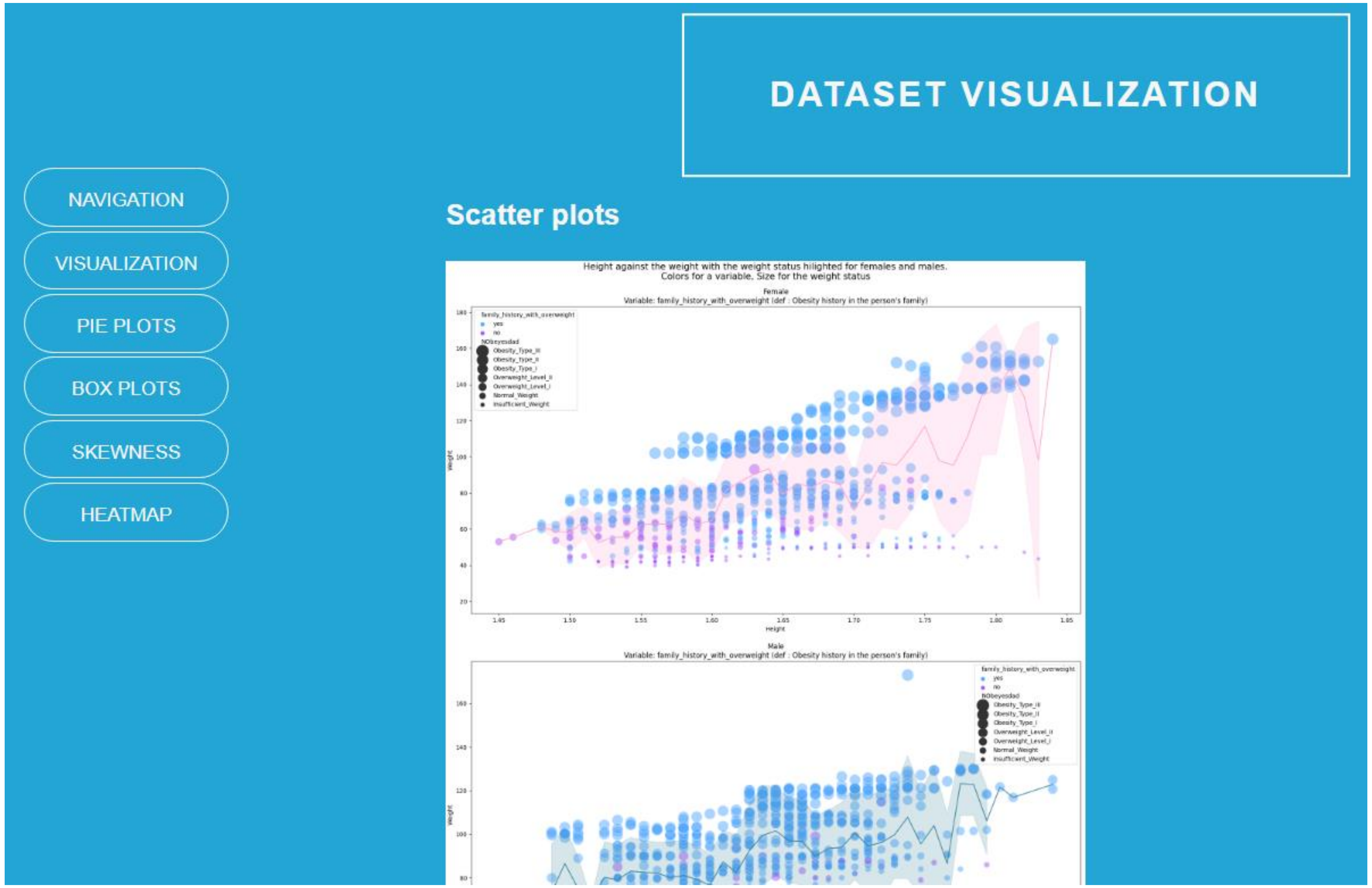
DATASET VISUALIZATION

Skewness

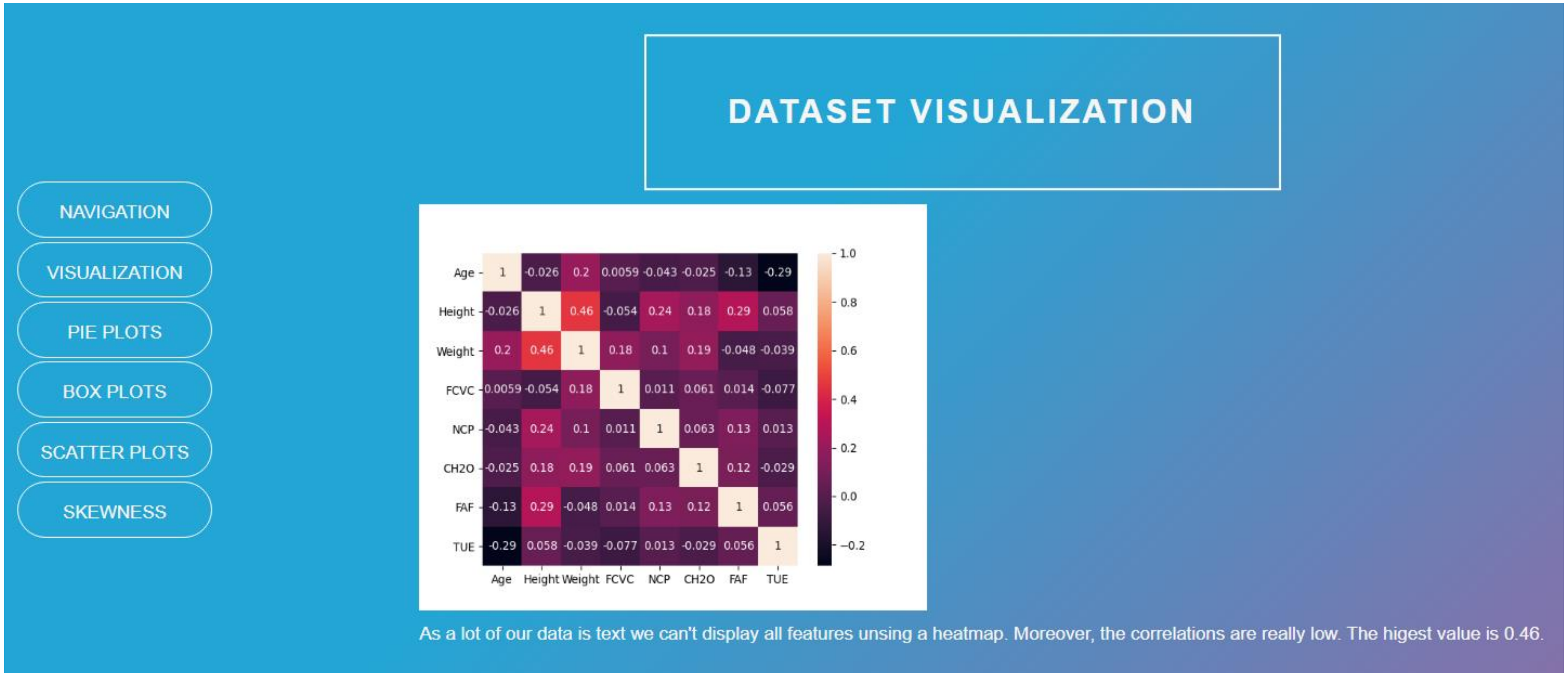
To calculate the skewness of a feature, we are using the following module : from scipy.stats import skew

Data is Positively or Right skewed. Skewness value: 1.520244920562755

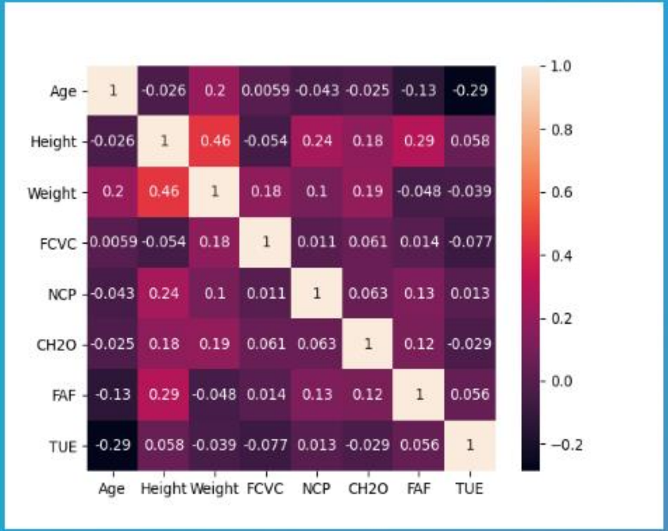
ANALYSIS


DATASET VISUALIZATION
NAVIGATION
VISUALIZATION
PIE PLOTS
BOX PLOTS
SKEWNESS
HEATMAP
Scatter plots
Height against the weight with the weight status hilighted for females and males.
Colors for a variable. Size for the weight status
Female
Variable: family_history_with_overweight (def : Obesity history in the person's family)
family_history_with_overweight
yes
no
NObeyesdad
Obesity_Type_III
Obesity_Type_II
Obesity_Type_I
Overweight_Level_II
Overweight_Level_I
Normal_Weight
Insufficient_Weight
Weight
Height
Male
Variable: family_history_with_overweight (def : Obesity history in the person's family)

ANALYSIS



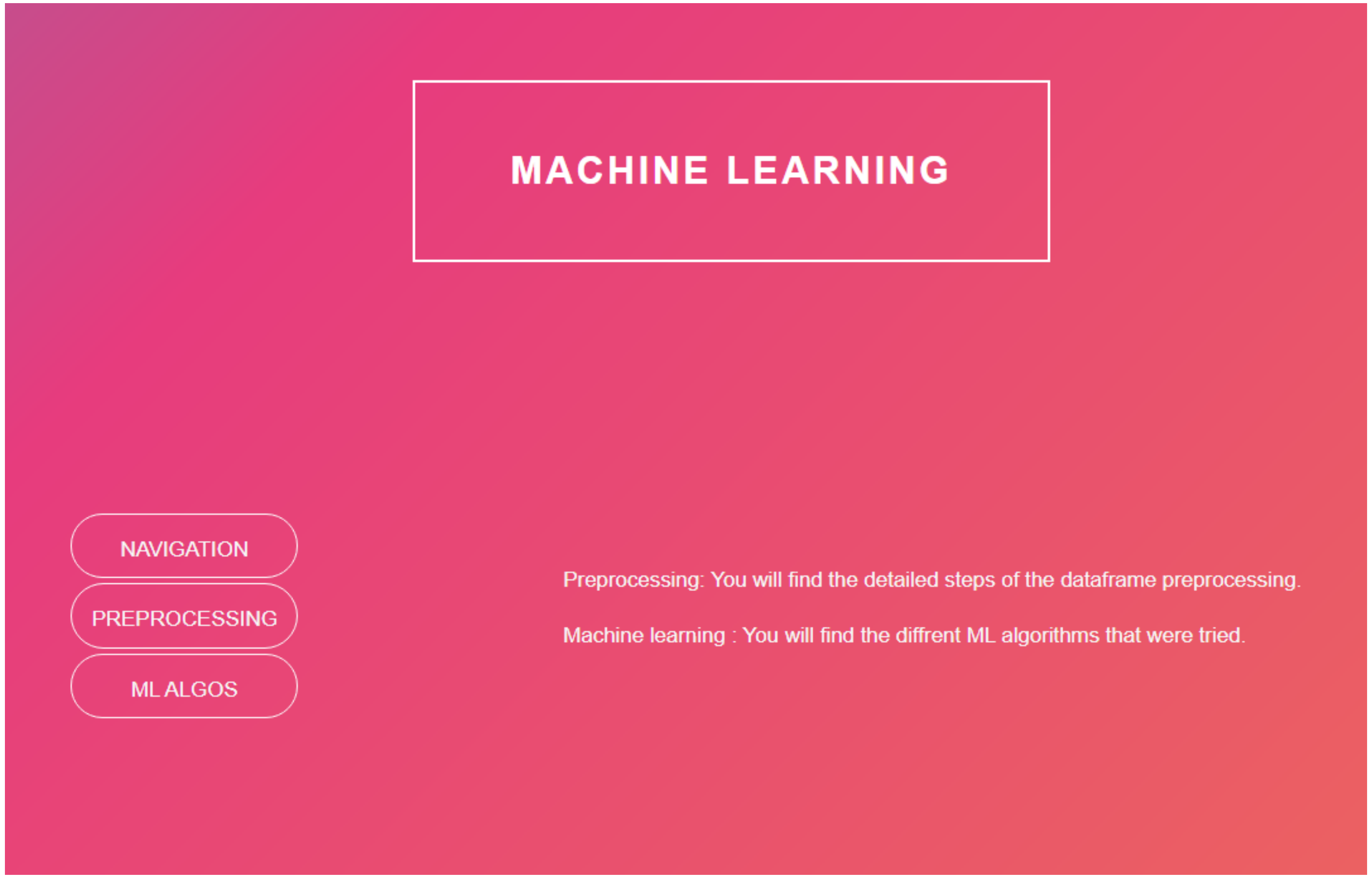
DATASET VISUALIZATION



	Age	Height	Weight	FCVC	NCP	CH2O	FAF	TUE
Age	1	-0.026	0.2	0.0059	-0.043	-0.025	-0.13	-0.29
Height	-0.026	1	0.46	-0.054	0.24	0.18	0.29	0.058
Weight	0.2	0.46	1	0.18	0.1	0.19	-0.048	-0.039
FCVC	0.0059	-0.054	0.18	1	0.011	0.061	0.014	-0.077
NCP	-0.043	0.24	0.1	0.011	1	0.063	0.13	0.013
CH2O	-0.025	0.18	0.19	0.061	0.063	1	0.12	-0.029
FAF	-0.13	0.29	-0.048	0.014	0.13	0.12	1	0.056
TUE	-0.29	0.058	-0.039	-0.077	0.013	-0.029	0.056	1

As a lot of our data is text we can't display all features unsing a heatmap. Moreover, the correlations are really low. The higest value is 0.46.

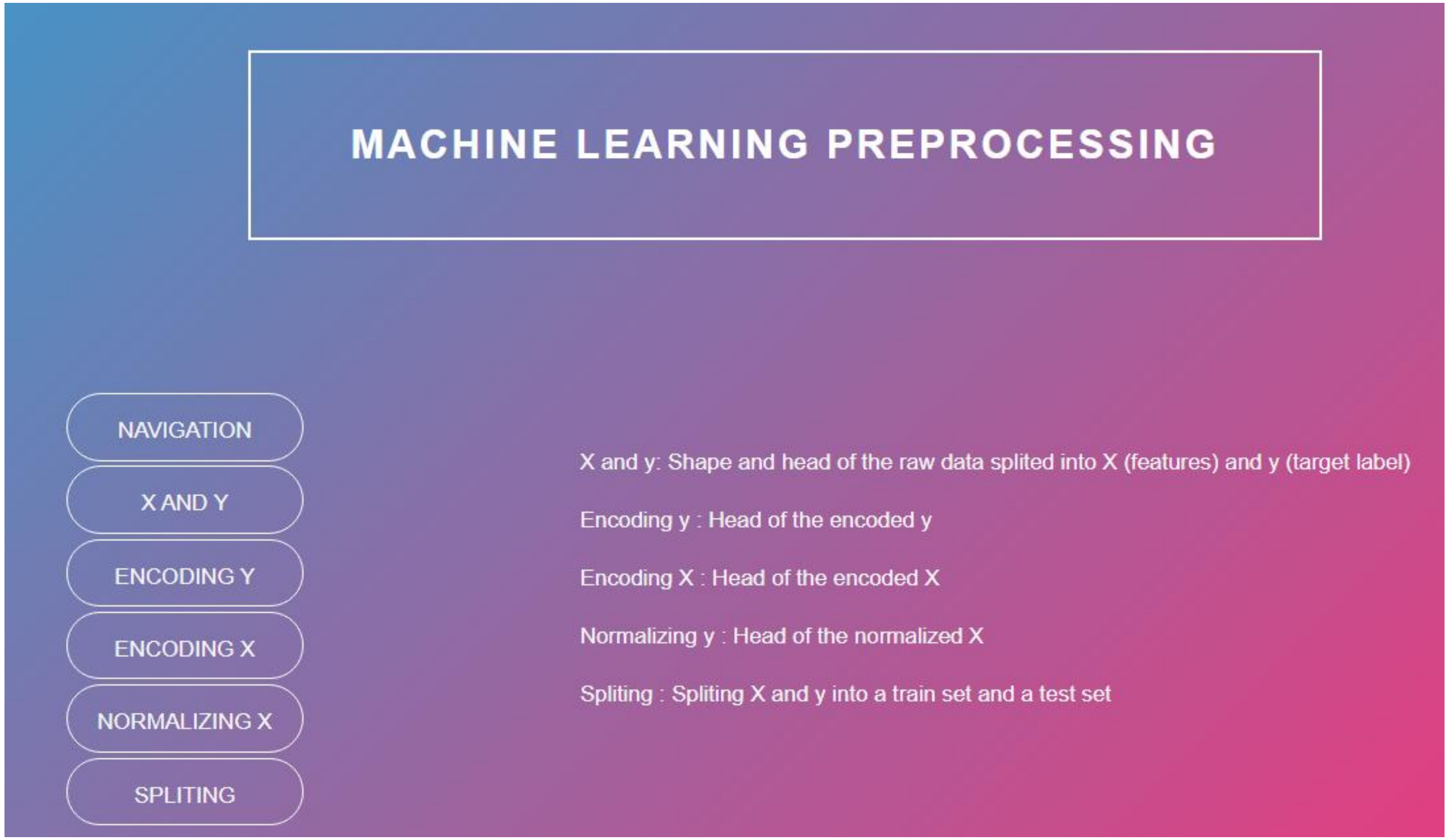
ML



Preprocessing: You will find the detailed steps of the dataframe preprocessing.

Machine learning : You will find the diffrent ML algorithms that were tried.

ML


MACHINE LEARNING PREPROCESSING

X and y: Shape and head of the raw data splited into X (features) and y (target label)

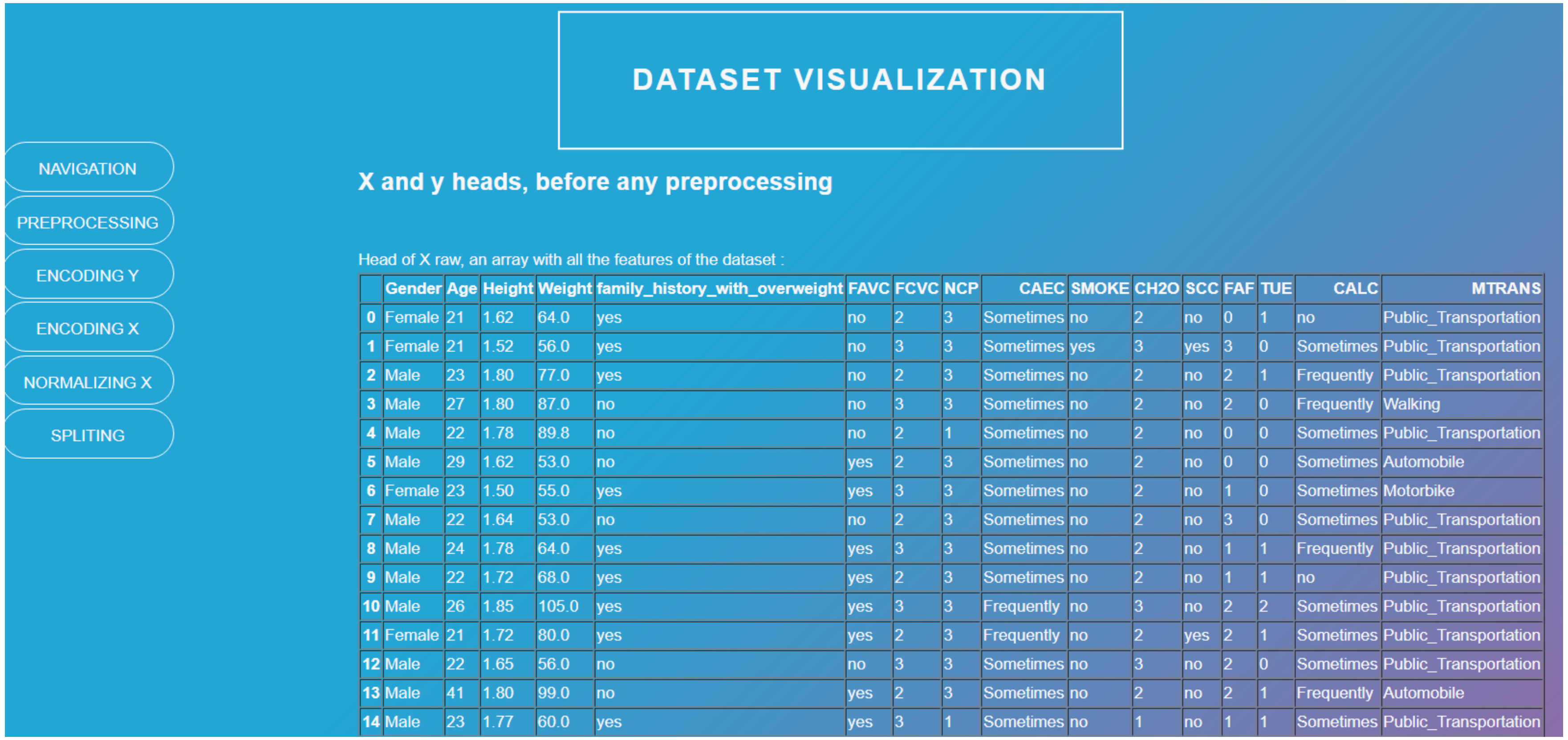
Encoding y : Head of the encoded y

Encoding X : Head of the encoded X

Normalizing y : Head of the normalized X

Spliting : Spliting X and y into a train set and a test set

M L



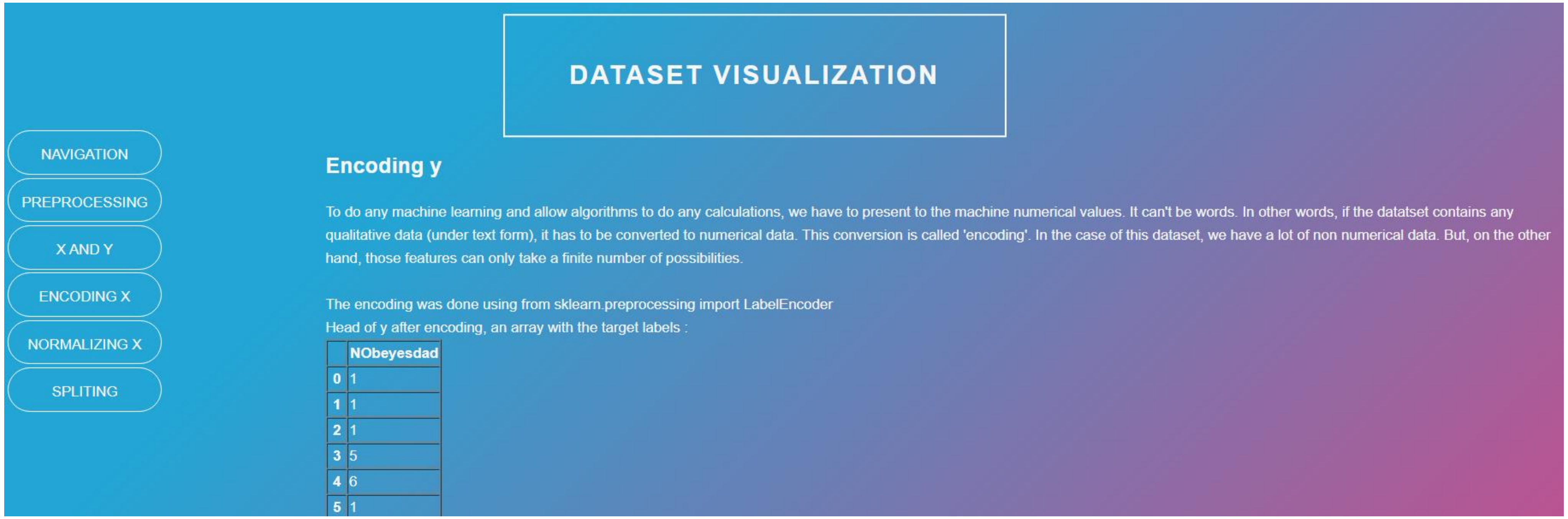
DATASET VISUALIZATION

X and y heads, before any preprocessing

Head of X raw, an array with all the features of the dataset :

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS
0	Female	21	1.62	64.0	yes	no	2	3	Sometimes	no	2	no	0	1	no	Public_Transportation
1	Female	21	1.52	56.0	yes	no	3	3	Sometimes	yes	3	yes	3	0	Sometimes	Public_Transportation
2	Male	23	1.80	77.0	yes	no	2	3	Sometimes	no	2	no	2	1	Frequently	Public_Transportation
3	Male	27	1.80	87.0	no	no	3	3	Sometimes	no	2	no	2	0	Frequently	Walking
4	Male	22	1.78	89.8	no	no	2	1	Sometimes	no	2	no	0	0	Sometimes	Public_Transportation
5	Male	29	1.62	53.0	no	yes	2	3	Sometimes	no	2	no	0	0	Sometimes	Automobile
6	Female	23	1.50	55.0	yes	yes	3	3	Sometimes	no	2	no	1	0	Sometimes	Motorbike
7	Male	22	1.64	53.0	no	no	2	3	Sometimes	no	2	no	3	0	Sometimes	Public_Transportation
8	Male	24	1.78	64.0	yes	yes	3	3	Sometimes	no	2	no	1	1	Frequently	Public_Transportation
9	Male	22	1.72	68.0	yes	yes	2	3	Sometimes	no	2	no	1	1	no	Public_Transportation
10	Male	26	1.85	105.0	yes	yes	3	3	Frequently	no	3	no	2	2	Sometimes	Public_Transportation
11	Female	21	1.72	80.0	yes	yes	2	3	Frequently	no	2	yes	2	1	Sometimes	Public_Transportation
12	Male	22	1.65	56.0	no	no	3	3	Sometimes	no	3	no	2	0	Sometimes	Public_Transportation
13	Male	41	1.80	99.0	no	yes	2	3	Sometimes	no	2	no	2	1	Frequently	Automobile
14	Male	23	1.77	60.0	yes	yes	3	1	Sometimes	no	1	no	1	1	Sometimes	Public_Transportation

M L



DATASET VISUALIZATION

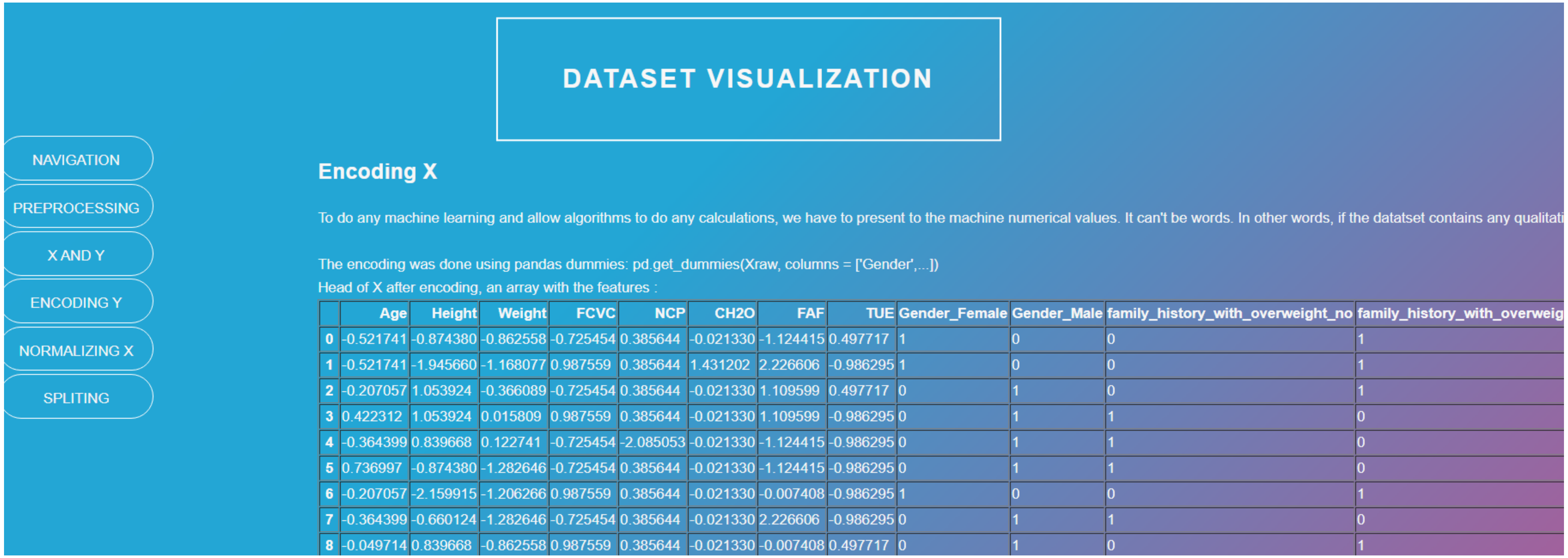
Encoding y

To do any machine learning and allow algorithms to do any calculations, we have to present to the machine numerical values. It can't be words. In other words, if the datatset contains any qualitative data (under text form), it has to be converted to numerical data. This conversion is called 'encoding'. In the case of this dataset, we have a lot of non numerical data. But, on the other hand, those features can only take a finite number of possibilities.

The encoding was done using from sklearn.preprocessing import LabelEncoder
Head of y after encoding, an array with the target labels :

	NObeyesdad
0	1
1	1
2	1
3	5
4	6
5	1

M L



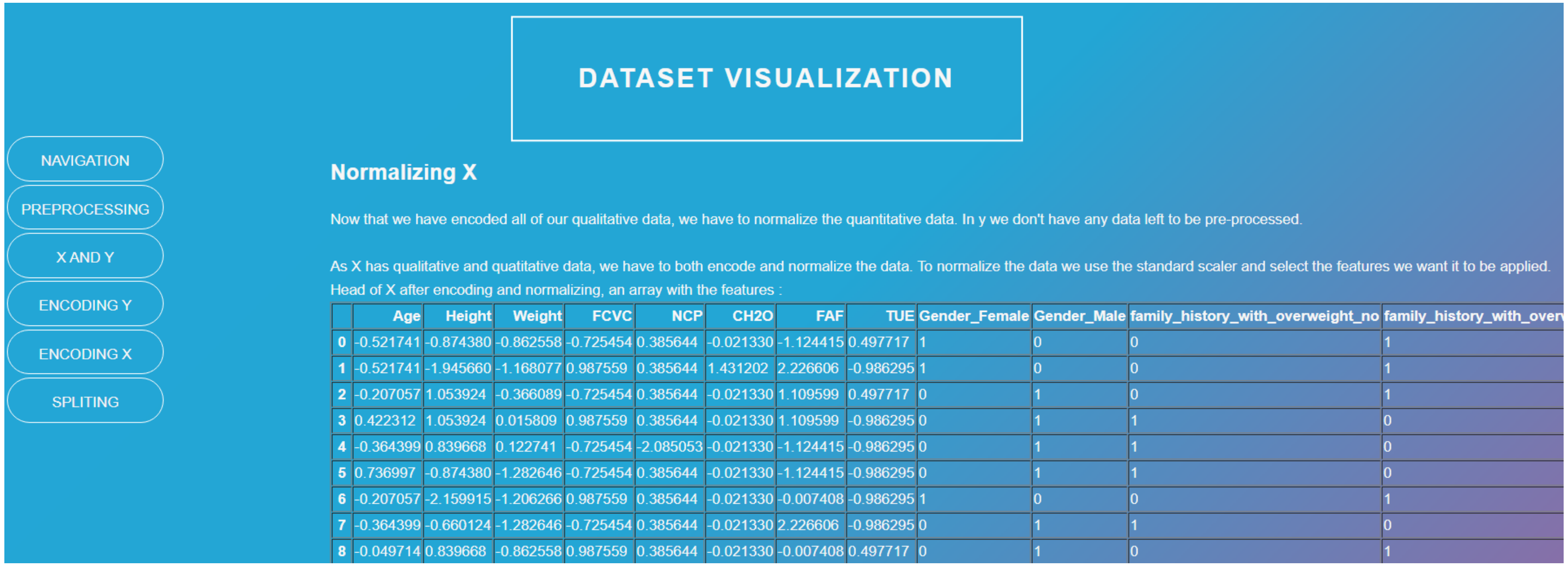
Encoding X

To do any machine learning and allow algorithms to do any calculations, we have to present to the machine numerical values. It can't be words. In other words, if the datatset contains any qualitati

The encoding was done using pandas dummies: pd.get_dummies(Xraw, columns = ['Gender',...])
Head of X after encoding, an array with the features :

	Age	Height	Weight	FCVC	NCP	CH2O	FAF	TUE	Gender_Female	Gender_Male	family_history_with_overweight_no	family_history_with_overweig
0	-0.521741	-0.874380	-0.862558	-0.725454	0.385644	-0.021330	-1.124415	0.497717	1	0	0	1
1	-0.521741	-1.945660	-1.168077	0.987559	0.385644	1.431202	2.226606	-0.986295	1	0	0	1
2	-0.207057	1.053924	-0.366089	-0.725454	0.385644	-0.021330	1.109599	0.497717	0	1	0	1
3	0.422312	1.053924	0.015809	0.987559	0.385644	-0.021330	1.109599	-0.986295	0	1	1	0
4	-0.364399	0.839668	0.122741	-0.725454	-2.085053	-0.021330	-1.124415	-0.986295	0	1	1	0
5	0.736997	-0.874380	-1.282646	-0.725454	0.385644	-0.021330	-1.124415	-0.986295	0	1	1	0
6	-0.207057	-2.159915	-1.206266	0.987559	0.385644	-0.021330	-0.007408	-0.986295	1	0	0	1
7	-0.364399	-0.660124	-1.282646	-0.725454	0.385644	-0.021330	2.226606	-0.986295	0	1	1	0
8	-0.049714	0.839668	-0.862558	0.987559	0.385644	-0.021330	-0.007408	0.497717	0	1	0	1

M L



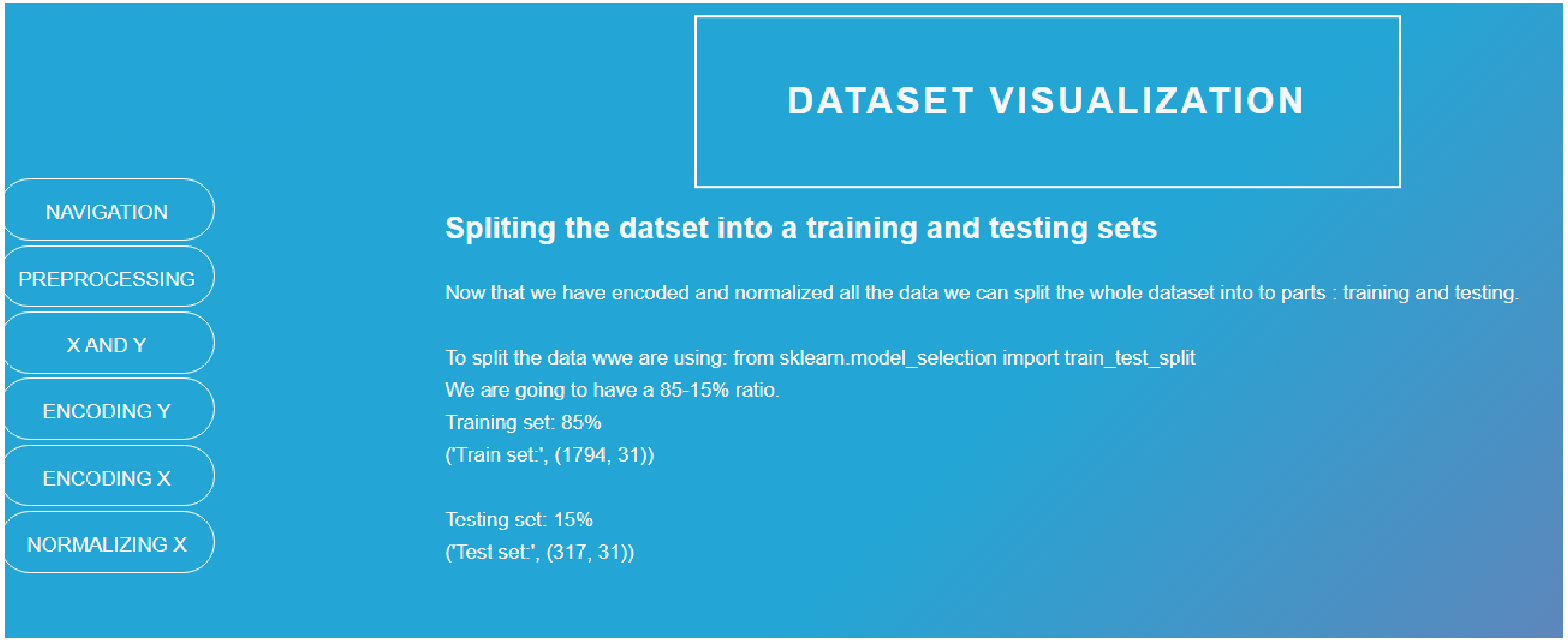
Normalizing X

Now that we have encoded all of our qualitative data, we have to normalize the quantitative data. In y we don't have any data left to be pre-processed.

As X has qualitative and quatitative data, we have to both encode and normalize the data. To normalize the data we use the standard scaler and select the features we want it to be applied.
Head of X after encoding and normalizing, an array with the features :

	Age	Height	Weight	FCVC	NCP	CH2O	FAF	TUE	Gender_Female	Gender_Male	family_history_with_overweight_no	family_history_with_over
0	-0.521741	-0.874380	-0.862558	-0.725454	0.385644	-0.021330	-1.124415	0.497717	1	0	0	1
1	-0.521741	-1.945660	-1.168077	0.987559	0.385644	1.431202	2.226606	-0.986295	1	0	0	1
2	-0.207057	1.053924	-0.366089	-0.725454	0.385644	-0.021330	1.109599	0.497717	0	1	0	1
3	0.422312	1.053924	0.015809	0.987559	0.385644	-0.021330	1.109599	-0.986295	0	1	1	0
4	-0.364399	0.839668	0.122741	-0.725454	-2.085053	-0.021330	-1.124415	-0.986295	0	1	1	0
5	0.736997	-0.874380	-1.282646	-0.725454	0.385644	-0.021330	-1.124415	-0.986295	0	1	1	0
6	-0.207057	-2.159915	-1.206266	0.987559	0.385644	-0.021330	-0.007408	-0.986295	1	0	0	1
7	-0.364399	-0.660124	-1.282646	-0.725454	0.385644	-0.021330	2.226606	-0.986295	0	1	1	0
8	-0.049714	0.839668	-0.862558	0.987559	0.385644	-0.021330	-0.007408	0.497717	0	1	0	1

ML



DATASET VISUALIZATION

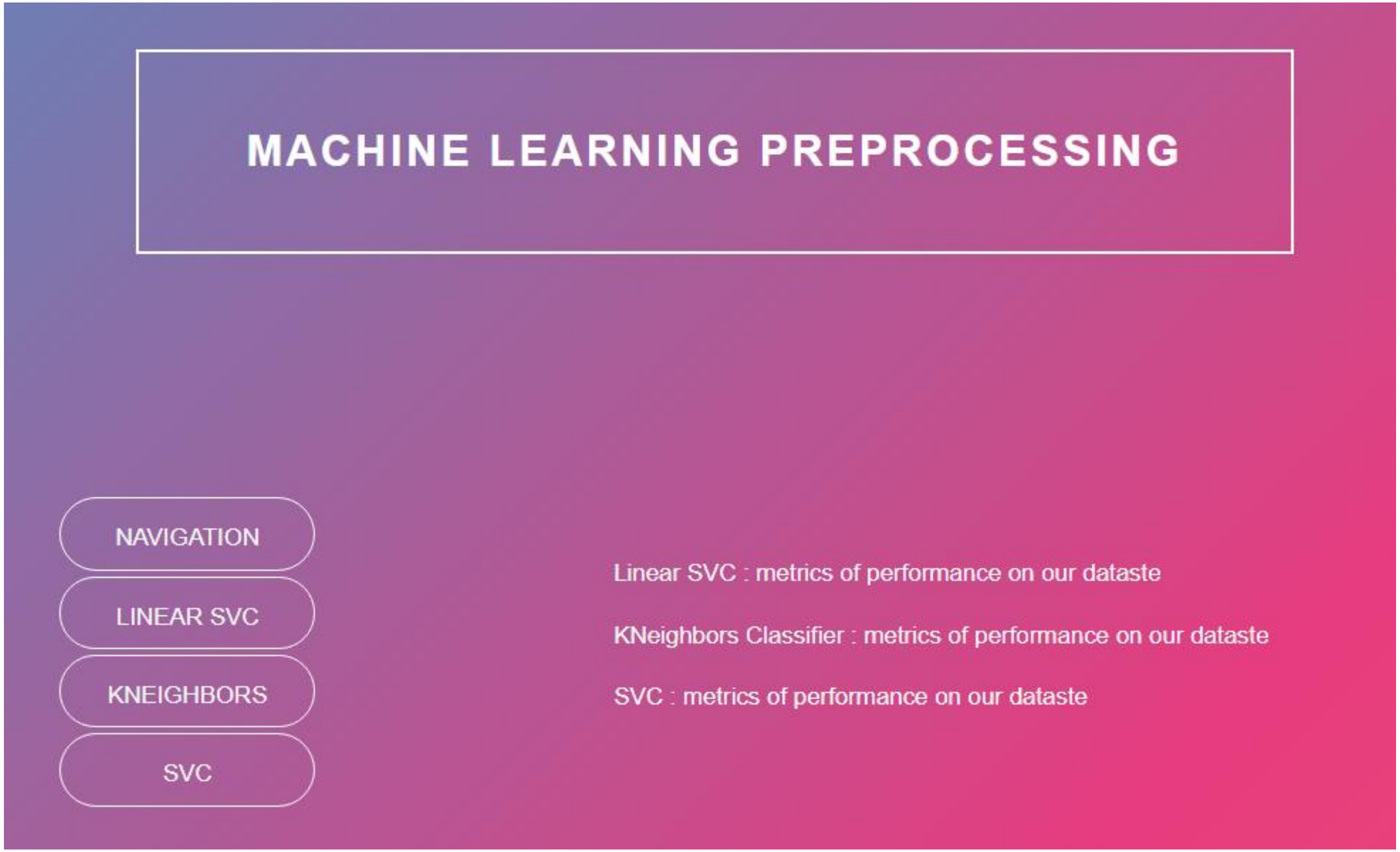
Spliting the datset into a training and testing sets

Now that we have encoded and normalized all the data we can split the whole dataset into to parts : training and testing.

To split the data wwe are using: from sklearn.model_selection import train_test_split
We are going to have a 85-15% ratio.
Training set: 85%
('Train set:', (1794, 31))

Testing set: 15%
('Test set:', (317, 31))

M L



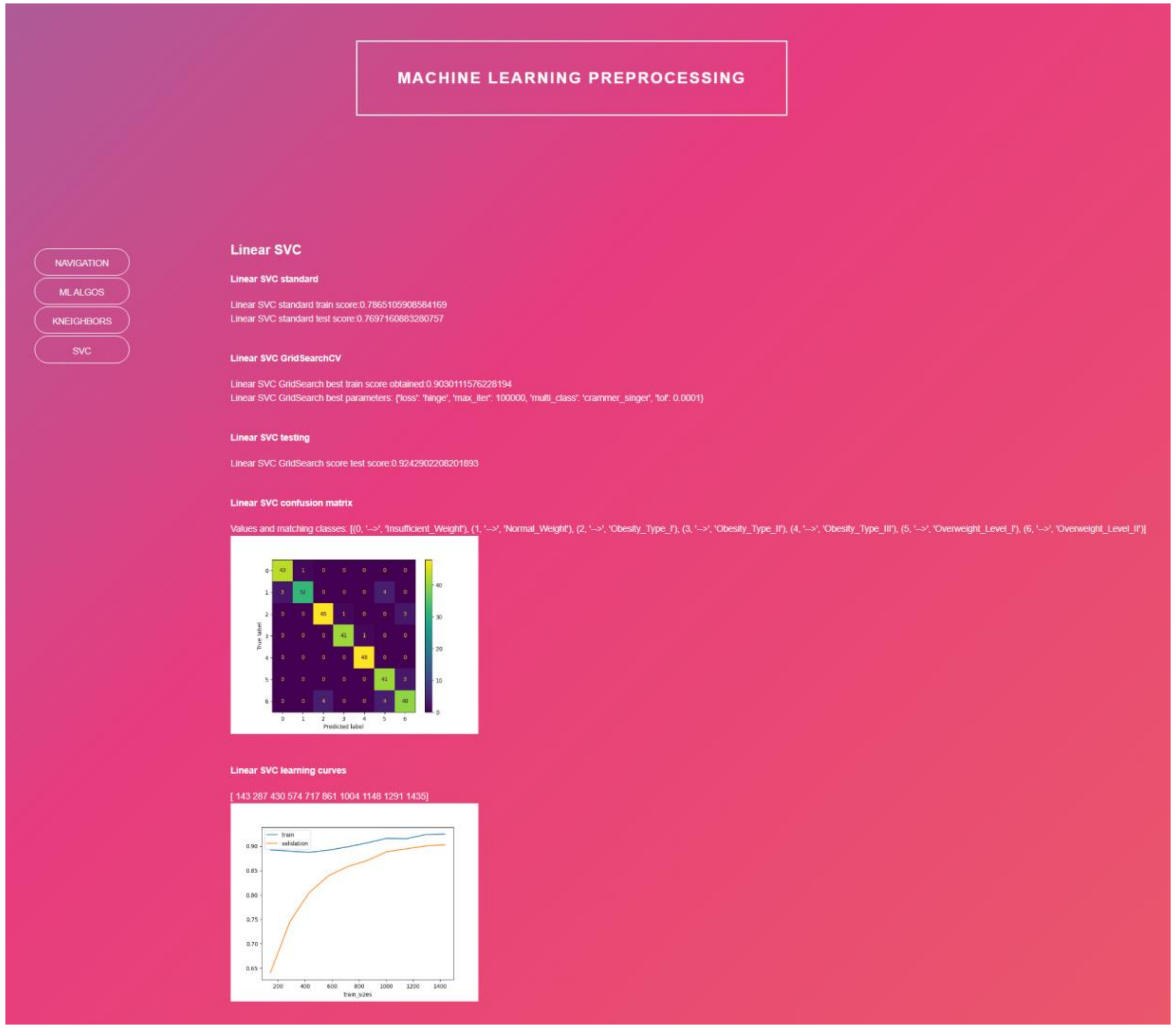
MACHINE LEARNING PREPROCESSING

Linear SVC : metrics of performance on our dataste

KNeighbors Classifier : metrics of performance on our dataste

SVC : metrics of performance on our dataste

LINEAR SVC



Linear SVC

Linear SVC standard

Linear SVC standard train score:0.7865105908584169
Linear SVC standard test score:0.7697160883280757

Linear SVC GridSearchCV

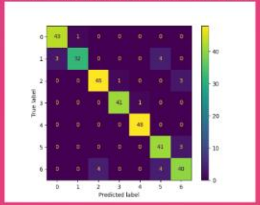
Linear SVC GridSearch best train score obtained:0.9030111576228194
Linear SVC GridSearch best parameters: {'loss': 'hinge', 'max_iter': 100000, 'multi_class': 'crammer_singer', 'tol': 0.0001}

Linear SVC testing

Linear SVC GridSearch score test score:0.9242902208201893

Linear SVC confusion matrix

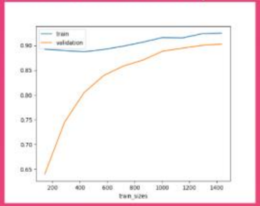
Values and matching classes: [(0, '-->', 'Insufficient_Weight'), (1, '-->', 'Normal_Weight'), (2, '-->', 'Obesity_Type_I'), (3, '-->', 'Obesity_Type_II'), (4, '-->', 'Obesity_Type_III'), (5, '-->', 'Overweight_Level_I'), (6, '-->', 'Overweight_Level_II')]



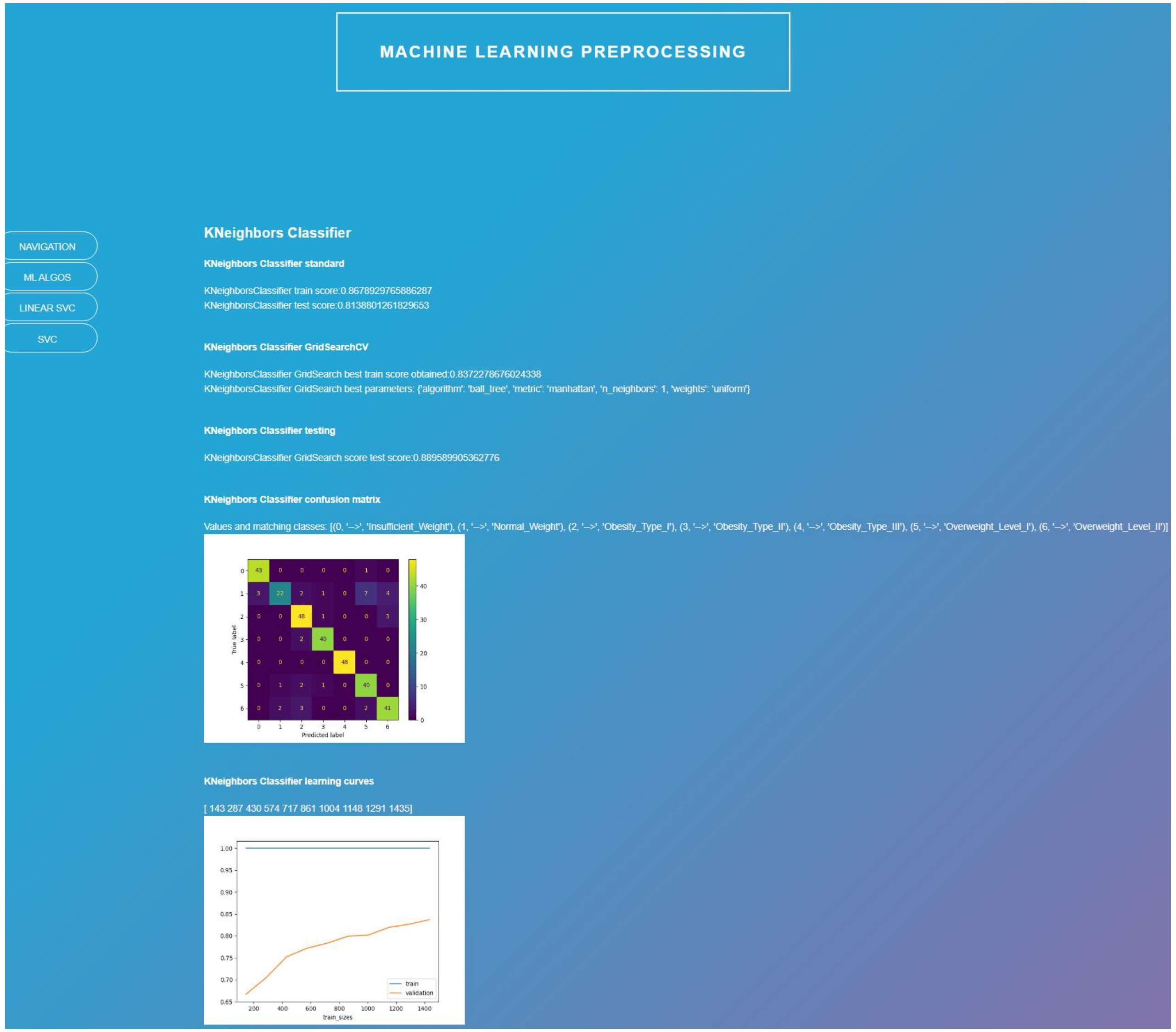
True label \ Predicted label	0	1	2	3	4	5	6
0	43	1	0	0	0	0	0
1	3	32	0	0	0	4	0
2	0	0	46	1	0	0	3
3	0	0	0	41	1	0	0
4	0	0	0	0	48	0	0
5	0	0	0	0	0	41	3
6	0	0	4	0	0	4	40

Linear SVC learning curves

[143 287 430 574 717 861 1004 1148 1291 1435]



KNEIHBORS CLASSIFIER



KNeighbors Classifier

KNeighbors Classifier standard

KNeighborsClassifier train score:0.8678929765886287
KNeighborsClassifier test score:0.8138801261829653

KNeighbors Classifier GridSearchCV

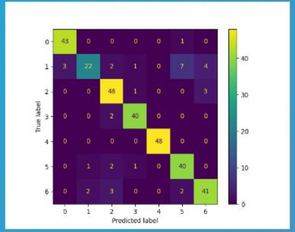
KNeighborsClassifier GridSearch best train score obtained:0.8372278676024338
KNeighborsClassifier GridSearch best parameters: {'algorithm': 'ball_tree', 'metric': 'manhattan', 'n_neighbors': 1, 'weights': 'uniform'}

KNeighbors Classifier testing

KNeighborsClassifier GridSearch score test score:0.889589905362776

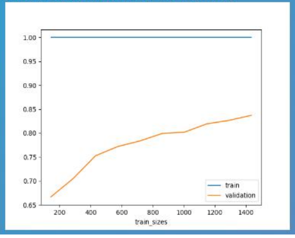
KNeighbors Classifier confusion matrix

Values and matching classes: [(0, '-->', 'Insufficient_Weight'), (1, '-->', 'Normal_Weight'), (2, '-->', 'Obesity_Type_I'), (3, '-->', 'Obesity_Type_II'), (4, '-->', 'Obesity_Type_III'), (5, '-->', 'Overweight_Level_I'), (6, '-->', 'Overweight_Level_II')]

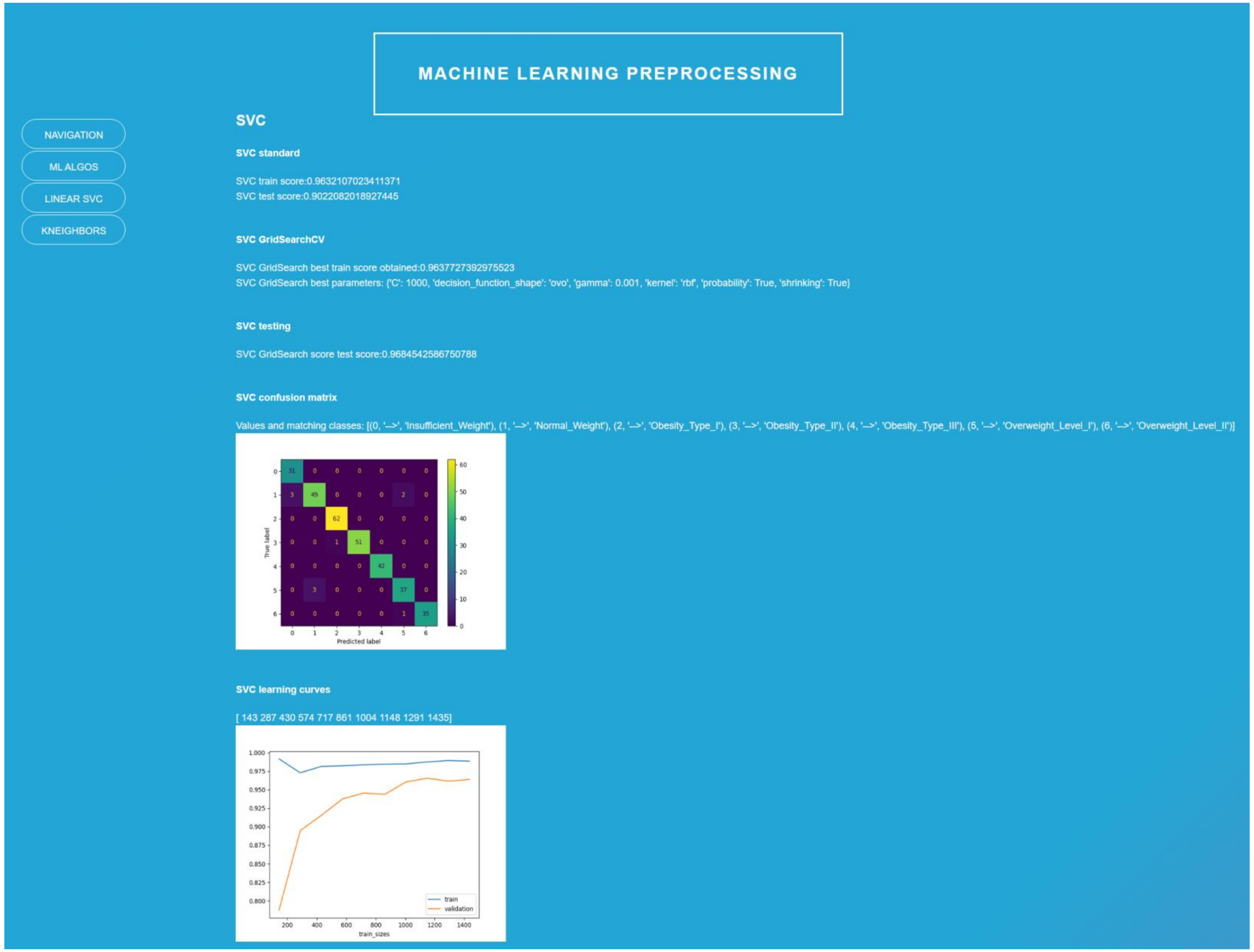


KNeighbors Classifier learning curves

[143 287 430 574 717 861 1004 1148 1291 1435]



SVC



MACHINE LEARNING PREPROCESSING

NAVIGATION
ML ALGOS
LINEAR SVC
KNEIGHBORS

SVC

SVC standard

SVC train score:0.9632107023411371
SVC test score:0.9022082018927445

SVC GridSearchCV

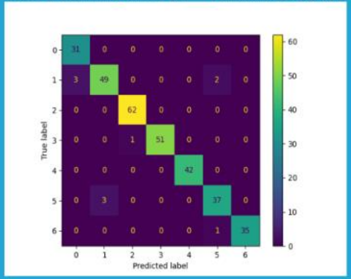
SVC GridSearch best train score obtained:0.9637727392975523
SVC GridSearch best parameters: {'C': 1000, 'decision_function_shape': 'ovo', 'gamma': 0.001, 'kernel': 'rbf', 'probability': True, 'shrinking': True}

SVC testing

SVC GridSearch score test score:0.9684542586750788

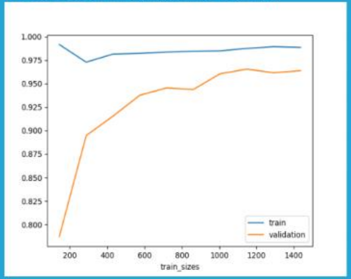
SVC confusion matrix

Values and matching classes: [(0, '-->', 'Insufficient_Weight'), (1, '-->', 'Normal_Weight'), (2, '-->', 'Obesity_Type_I'), (3, '-->', 'Obesity_Type_II'), (4, '-->', 'Obesity_Type_III'), (5, '-->', 'Overweight_Level_I'), (6, '-->', 'Overweight_Level_II')]



SVC learning curves

[143 287 430 574 717 861 1004 1148 1291 1435]



Python for Data analysis