République Tunisienne

Ministère de l'Enseinement Superieur et de la Recherche Scientifique

****

Institut Superieur d'Informatique et de Mathématiques de Monastir

Département d'informatique

Project Name:

Cardio Vascular Disease Detection

Team:

Omar kassar

&

Yosr mdemagh

&

Ala kerkeni

Class:

ING1_INFO_TD2

Academic year:

2023/2024

1.Data Engineering

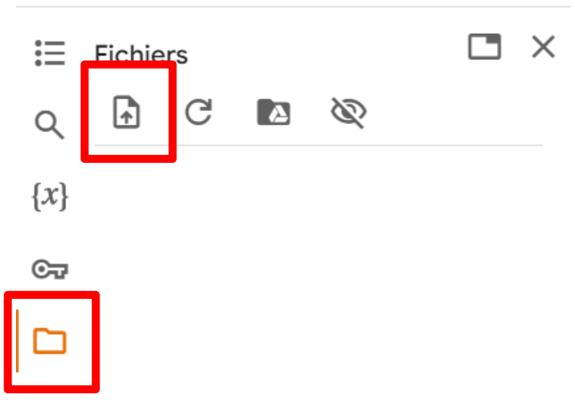
Question 1.a , 1.b and 1.c:

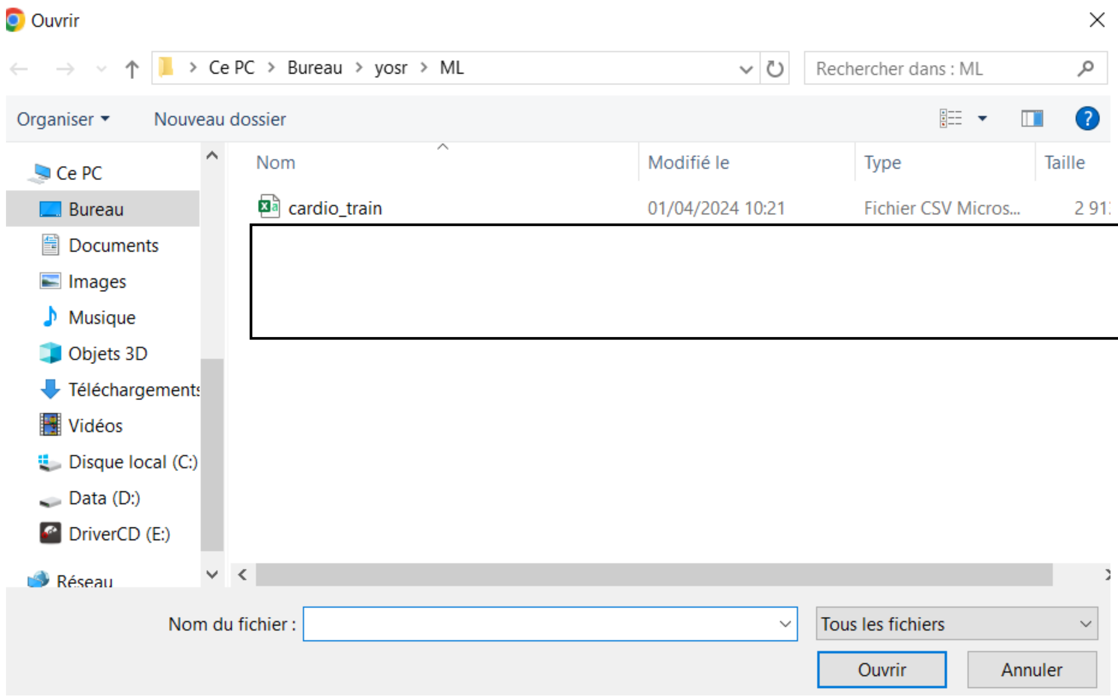
- Download the data and report how you prepare the data for machine learning:
- Load and represent the data using an appropriate data structure.
- Give a description of the data set like size, features, target variables, predictive variables, feature types, etc

In this step, we are going to explore our dataset in order to understand it and gain insights. The steps that we have taken are ordered as follows:

✓ **Importing the dataset:**

We used Google Colab. To work with the dataset, we need to import the CSV file:





✓ **Downloading and loading the dataset:**

```python
# Load data from CSV file into a DataFrame
df=pd.read_csv("/content/cardio_train.csv",sep=";")
```

⇨ We loaded data from a CSV file into a DataFrame (the data structure that we chose) using the Pandas library

✓ **Exploring columns:**

```python
df.columns
```

```
Index(['age', 'gender', 'ap_hi', 'ap_lo', 'cholesterol', 'gluc', 'smoke',
       'alco', 'active', 'cardio', 'BMI', 'BMI_Stage'],
      dtype='object')
```

✓ **Understanding the signification of each column:**

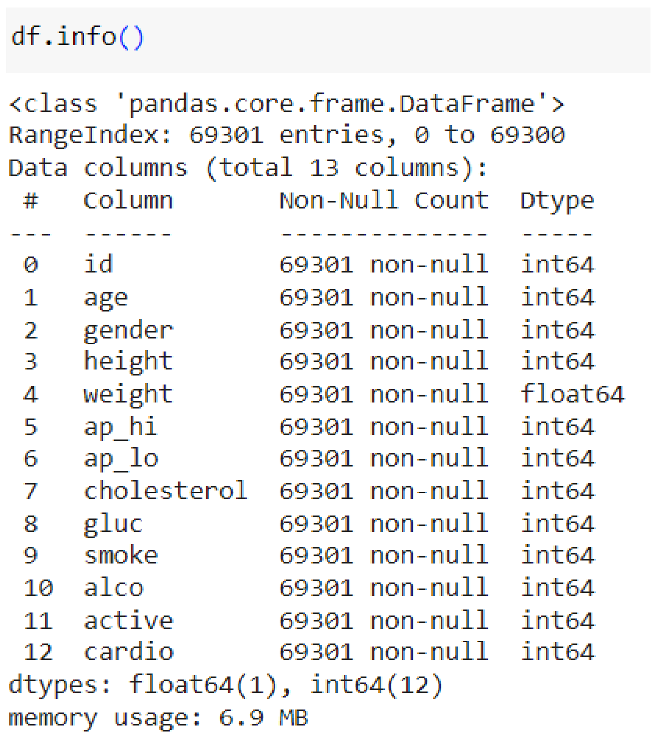
```
1.  Age | Objective Feature | age | int (days)
2.  Height | Objective Feature | height | int (cm) |
3.  Weight | Objective Feature | weight | float (kg) |
4.  Gender | Objective Feature | gender | categorical code |
5.  Systolic blood pressure | Examination Feature | ap_hi | int |
6.  Diastolic blood pressure | Examination Feature | ap_lo | int |
7.  Cholesterol | Examination Feature | cholesterol | 1: normal, 2:
above normal, 3: well above normal |
8.  Glucose | Examination Feature | gluc | 1: normal, 2: above normal,
3: well above normal |
9.  Smoking | Subjective Feature | smoke | binary |
10. Alcohol intake | Subjective Feature | alco | binary |
11. Physical activity | Subjective Feature | active | binary |
12. Presence or absence of cardiovascular disease | Target Variable |
cardio | binary |
```

✓ **displaying some rows of the dataset:**

```python
df.sample(10)
```

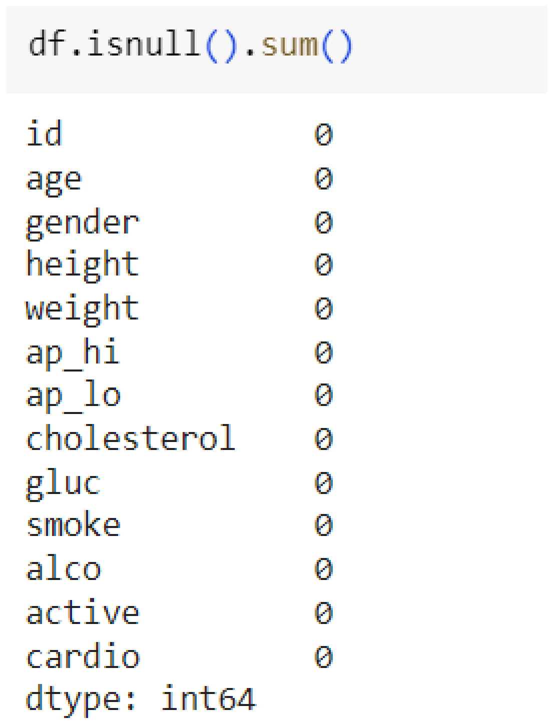
	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
36588	53259	18334	1	169	81.0	130	80	1	1	0	0	0	0
46955	68046	16676	1	163	65.0	120	80	1	1	0	0	1	0
2398	4368	21986	1	167	64.0	110	70	1	1	0	0	1	1
5150	8298	21732	1	156	80.0	120	80	1	1	0	0	1	1
69276	99963	21264	2	182	100.0	120	80	1	1	0	0	1	1
24498	35971	22645	2	182	90.0	130	80	3	1	0	0	1	1
33693	49141	19794	1	164	68.0	120	80	1	1	0	0	1	0
42748	62077	17452	2	170	92.0	120	80	1	1	0	0	1	0
11944	18049	17527	1	156	102.0	120	70	1	1	0	0	1	0

✓ **Observing general information about the dataset, such as the data type of each column, memory usage, and the presence of missing values**



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 69301 entries, 0 to 69300
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   id           69301 non-null  int64  
 1   age          69301 non-null  int64  
 2   gender       69301 non-null  int64  
 3   height       69301 non-null  int64  
 4   weight       69301 non-null  float64
 5   ap_hi        69301 non-null  int64  
 6   ap_lo        69301 non-null  int64  
 7   cholesterol  69301 non-null  int64  
 8   gluc         69301 non-null  int64  
 9   smoke        69301 non-null  int64  
 10  alco         69301 non-null  int64  
 11  active       69301 non-null  int64  
 12  cardio       69301 non-null  int64  
dtypes: float64(1), int64(12)
memory usage: 6.9 MB
```

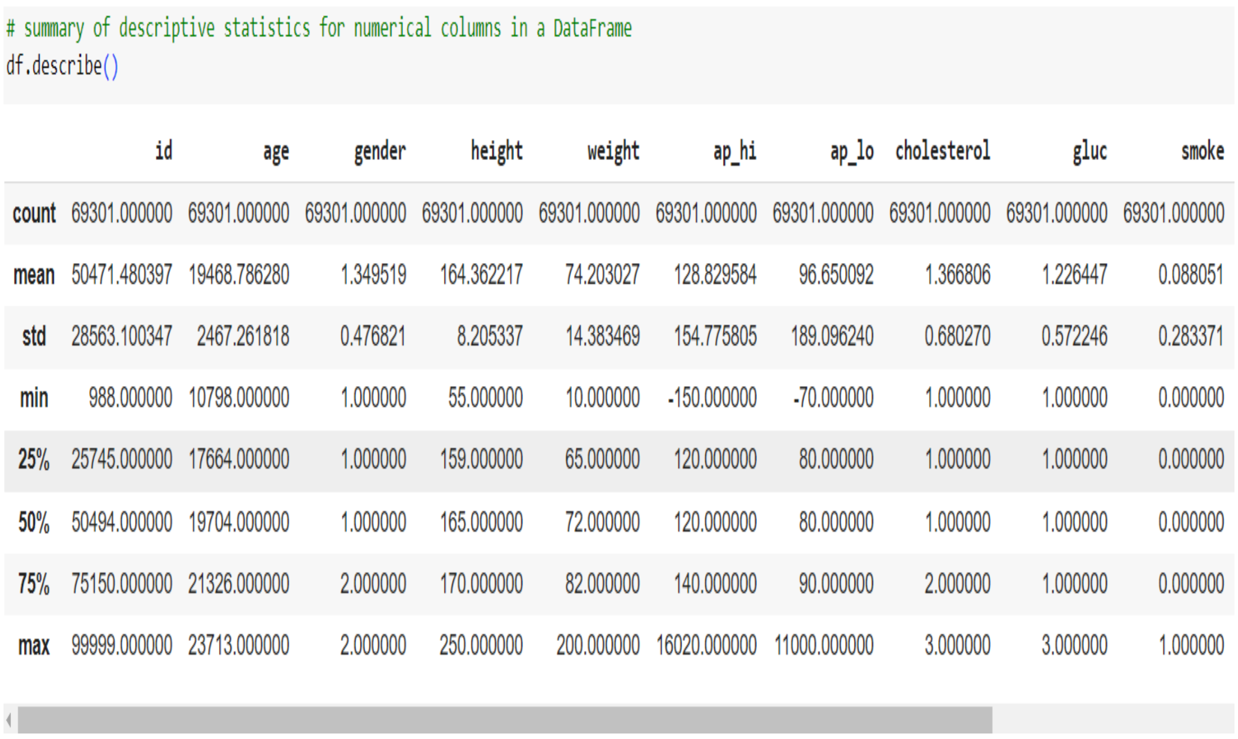


```
df.isnull().sum()
```

```
id             0
age            0
gender         0
height         0
weight         0
ap_hi          0
ap_lo          0
cholesterol    0
gluc           0
smoke          0
alco           0
active         0
cardio         0
dtype: int64
```

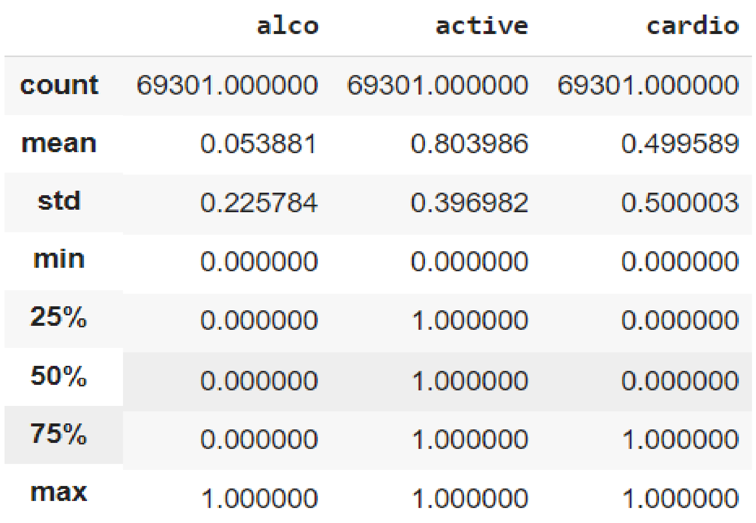
- ⇨ We have 12 column all of them are integer except weight is float
- ⇨ We don't have messing values
- ⇨ Memory usage is around 6.9 MB

✓ **Data description:**



```python
# summary of descriptive statistics for numerical columns in a DataFrame
df.describe()
```

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke
count	69301.000000	69301.000000	69301.000000	69301.000000	69301.000000	69301.000000	69301.000000	69301.000000	69301.000000	69301.000000
mean	50471.480397	19468.786280	1.349519	164.362217	74.203027	128.829584	96.650092	1.366806	1.226447	0.088051
std	28563.100347	2467.261818	0.476821	8.205337	14.383469	154.775805	189.096240	0.680270	0.572246	0.283371
min	988.000000	10798.000000	1.000000	55.000000	10.000000	-150.000000	-70.000000	1.000000	1.000000	0.000000
25%	25745.000000	17664.000000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000
50%	50494.000000	19704.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000
75%	75150.000000	21326.000000	2.000000	170.000000	82.000000	140.000000	90.000000	2.000000	1.000000	0.000000
max	99999.000000	23713.000000	2.000000	250.000000	200.000000	16020.000000	11000.000000	3.000000	3.000000	1.000000



	alco	active	cardio
count	69301.000000	69301.000000	69301.000000
mean	0.053881	0.803986	0.499589
std	0.225784	0.396982	0.500003
min	0.000000	0.000000	0.000000
25%	0.000000	1.000000	0.000000
50%	0.000000	1.000000	0.000000
75%	0.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000

For age (calculated in number of days lived):

Mean (=19468,78) = median (=19704) ➔we can say that we have a symmetric distribution

Min (=10798), max (=23713), std (=2476) ➔ low distribution

For height:

Mean (=164,36) = median (=165) ➔we can say that we have a symmetric distribution

Min (=55), max (=250), std (=8) ➔ low distribution

For weight:

Mean (=74,20) ~= median (=72) ➔we can say that we have a symmetric distribution

Min (=55), max (=250), std (=8) ➔ low distribution

For ap_hi:

Mean (=128,20) > median (=120) ➔ We can say that we have a positively skewed distribution

Min (=-150), max (=-16020), std (=147,55) ➔ low distribution

For ap_lo:

Mean (=96,65) > median (=80) ➔ We can say that we have a positively skewed distribution

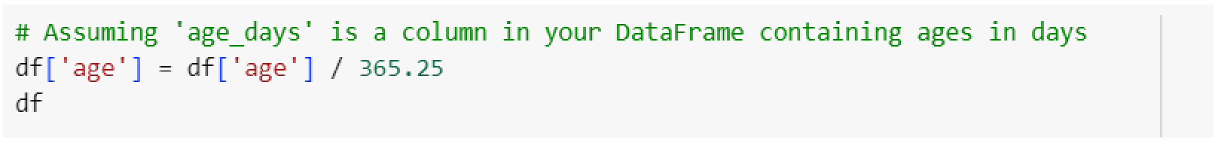
Min (=-70), max (=11000), std (=189) ➔ low distribution

Smoke, alcohol, active, cardio, gluc, and gender are categorical variables. We can better understand them through plots; description alone may not provide significant insights

Question 1.d:

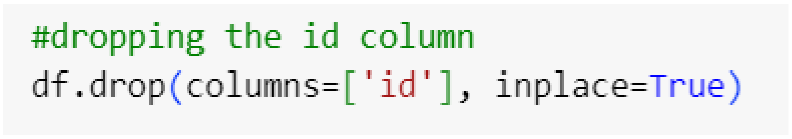
- **Apply any preprocessing steps that might be required to clean or filter the data before analysis**

✓ **converting the age from days to years for better readibility:**



```python
# Assuming 'age_days' is a column in your DataFrame containing ages in days
df['age'] = df['age'] / 365.25
df
```

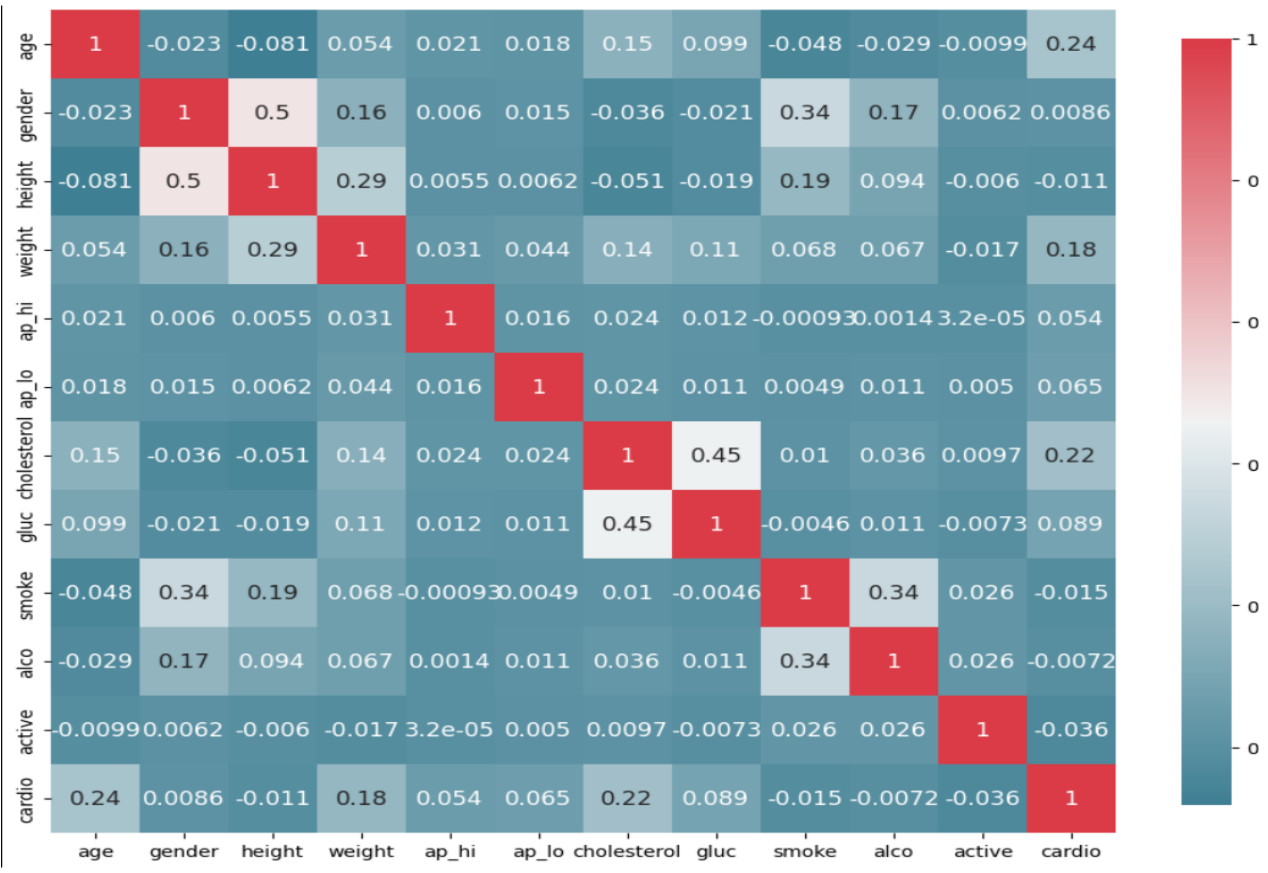
✓ **removing the id:**



```python
#dropping the id column
df.drop(columns=['id'], inplace=True)
```

Question 1.e an d 1.f:

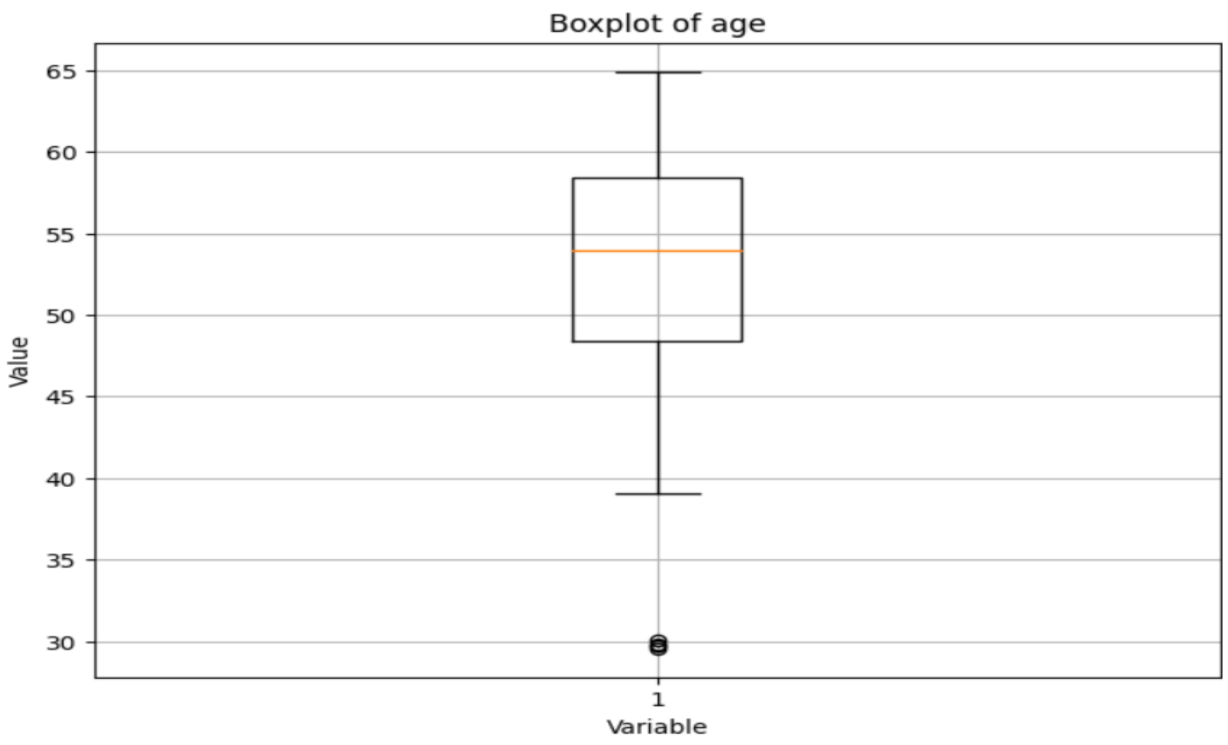
✓ **Verifying If there are corelated variables:**

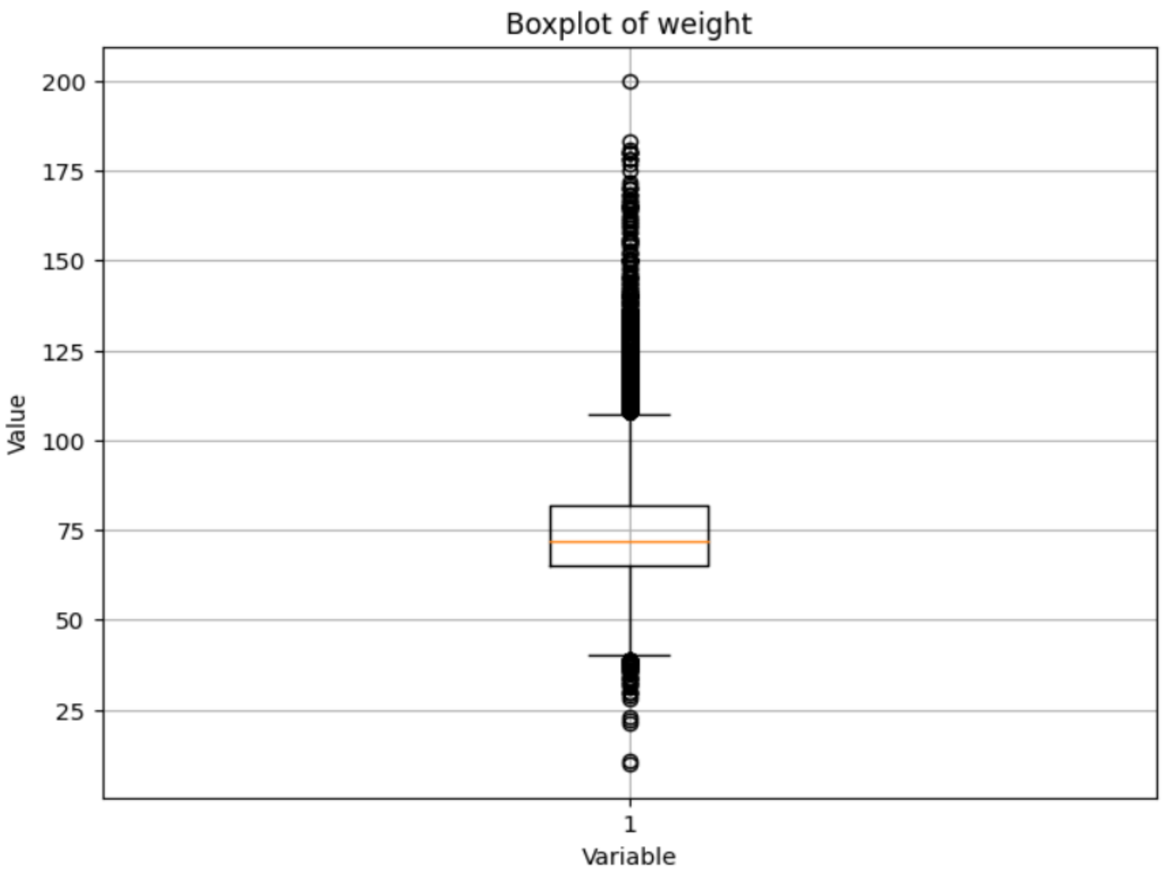


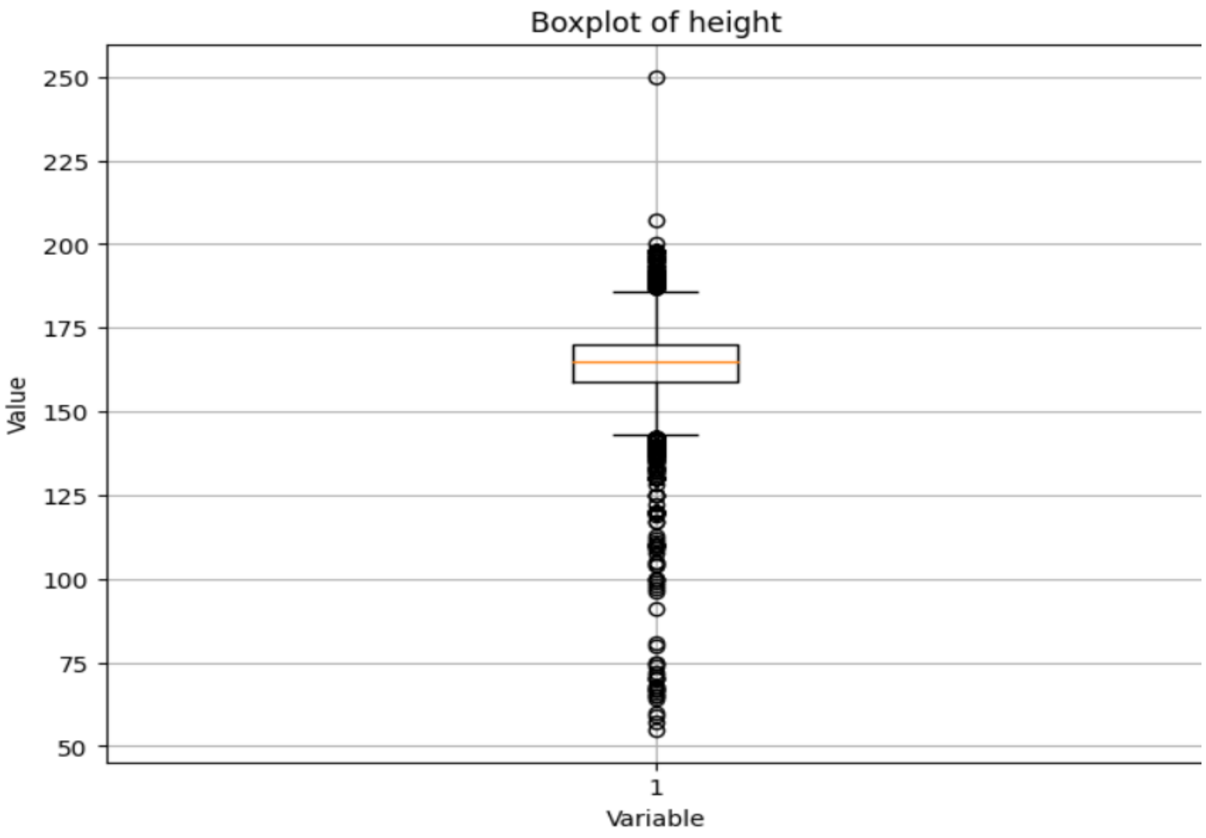
	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
age	1	-0.023	-0.081	0.054	0.021	0.018	0.15	0.099	-0.048	-0.029	-0.0099	0.24
gender	-0.023	1	0.5	0.16	0.006	0.015	-0.036	-0.021	0.34	0.17	0.0062	0.0086
height	-0.081	0.5	1	0.29	0.0055	0.0062	-0.051	-0.019	0.19	0.094	-0.006	-0.011
weight	0.054	0.16	0.29	1	0.031	0.044	0.14	0.11	0.068	0.067	-0.017	0.18
ap_hi	0.021	0.006	0.0055	0.031	1	0.016	0.024	0.012	-0.00093	0.0014	3.2e-05	0.054
ap_lo	0.018	0.015	0.0062	0.044	0.016	1	0.024	0.011	0.0049	0.011	0.005	0.065
cholesterol	0.15	-0.036	-0.051	0.14	0.024	0.024	1	0.45	0.01	0.036	0.0097	0.22
gluc	0.099	-0.021	-0.019	0.11	0.012	0.011	0.45	1	-0.0046	0.011	-0.0073	0.089
smoke	-0.048	0.34	0.19	0.068	-0.00093	0.0049	0.01	-0.0046	1	0.34	0.026	-0.015
alco	-0.029	0.17	0.094	0.067	0.0014	0.011	0.036	0.011	0.34	1	0.026	-0.0072
active	-0.0099	0.0062	-0.006	-0.017	3.2e-05	0.005	0.0097	-0.0073	0.026	0.026	1	-0.036
cardio	0.24	0.0086	-0.011	0.18	0.054	0.065	0.22	0.089	-0.015	-0.0072	-0.036	1

This correlation matrix shows that There isn't much correlation between variables; only gender and height show a slight correlation, as do cholesterol and glucose

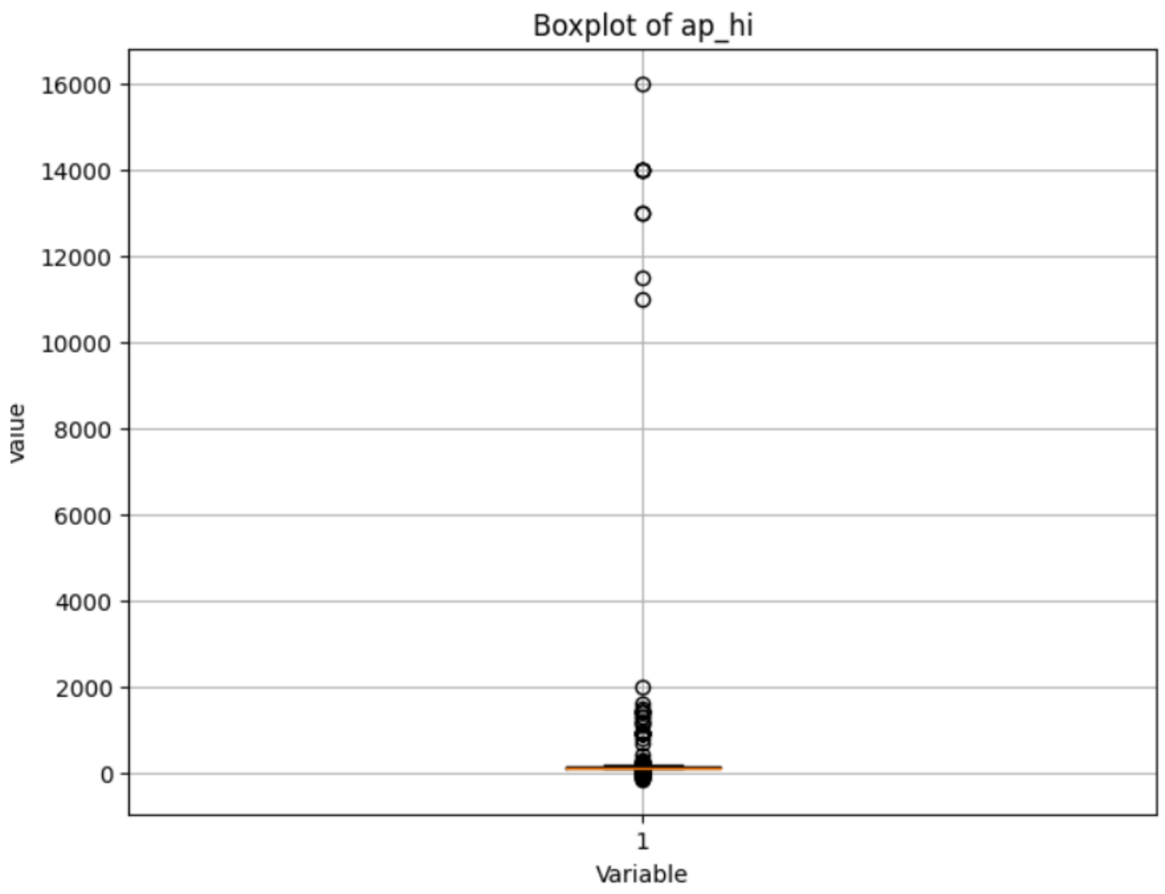
- ✓ **Visualizing data using box plots:**

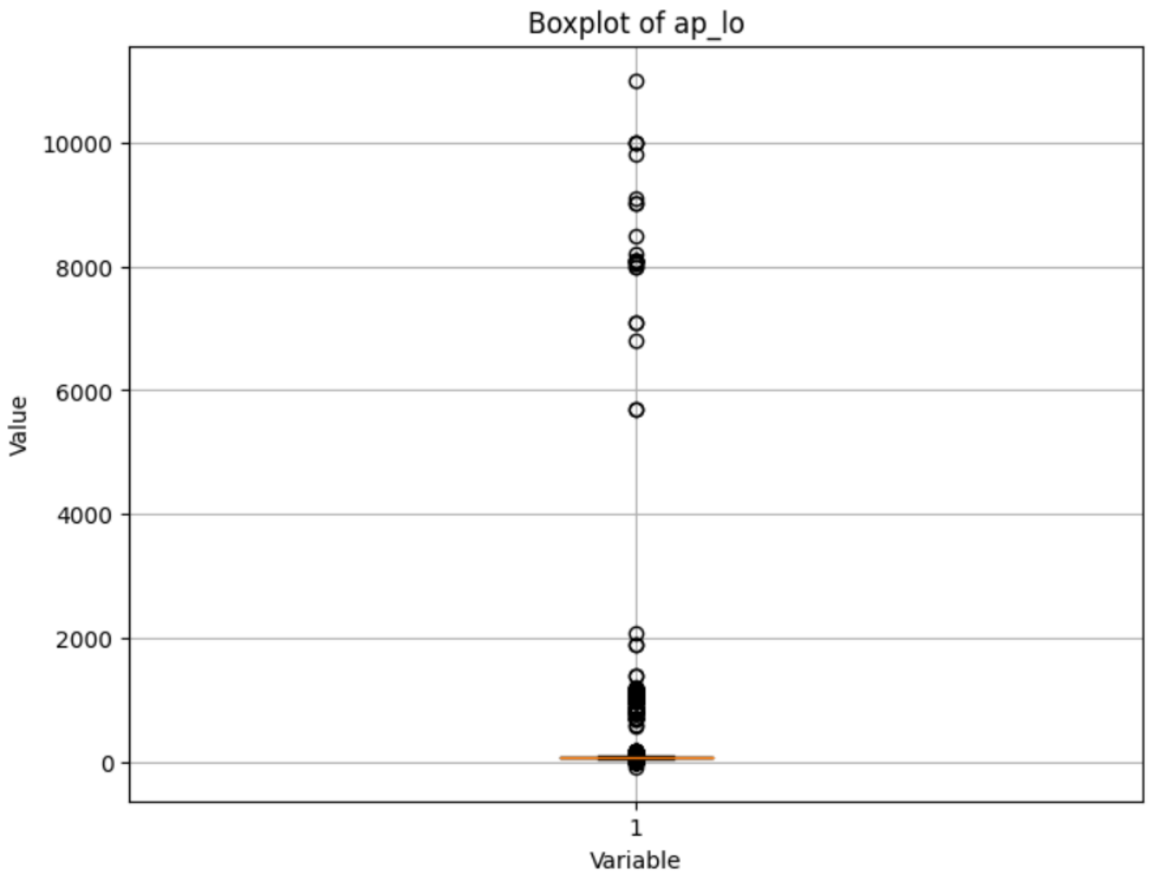




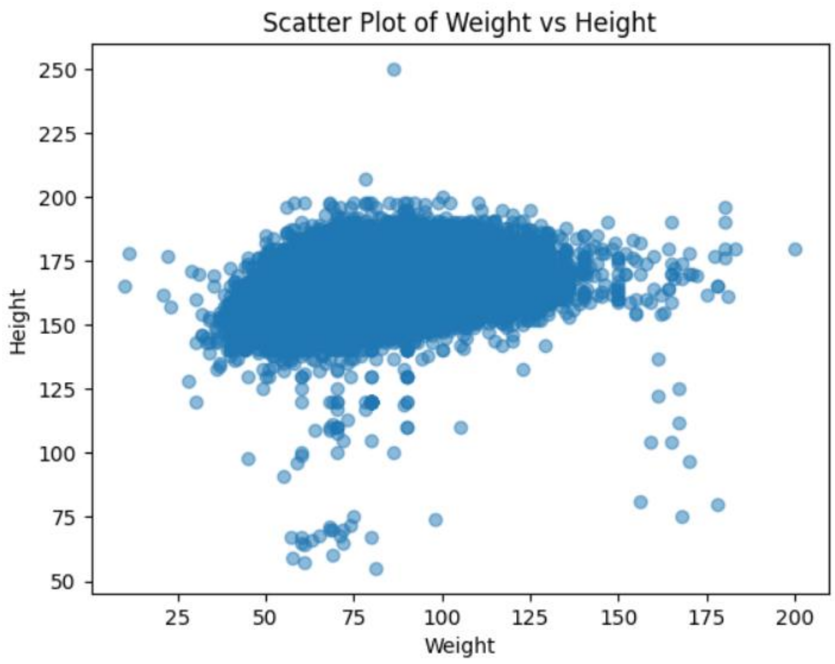


Team : omar kassar & yosr mdemagh & ala kerkeni

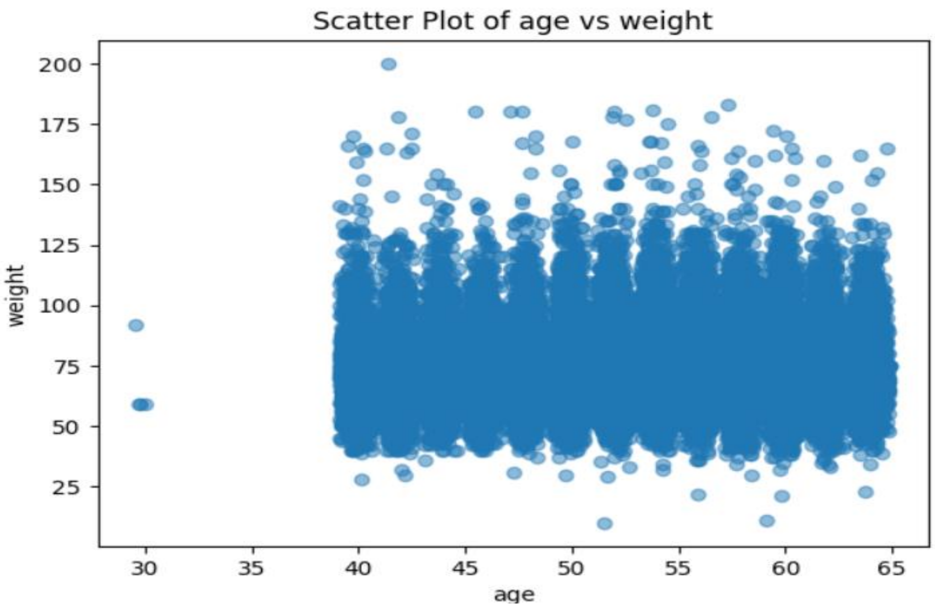

Boxplot of ap_hi
16000
14000
12000
10000
8000
6000
4000
2000
0
value
1
Variable


Boxplot of ap_lo
10000
8000
6000
4000
2000
0
Value
1
Variable

- ⇨ There are outliers in ap_lo , ap_hi, weight , height ,age.we have to deal with them before training our model
  - ✓ **Visualizing data using scatter plots:**



- ⇨ we can remak that nearly the majority of peaple have height between 125 cm and 290 and weight between 75 kg and 175 KG
- ⇨ there are outliers
- ⇨ ther isn't correlation between height and weight
- ⇨ there is only one cluster
- ⇨ there is no correlation between age and height
- ⇨ there is only one cluster
- ⇨ there are some outliers

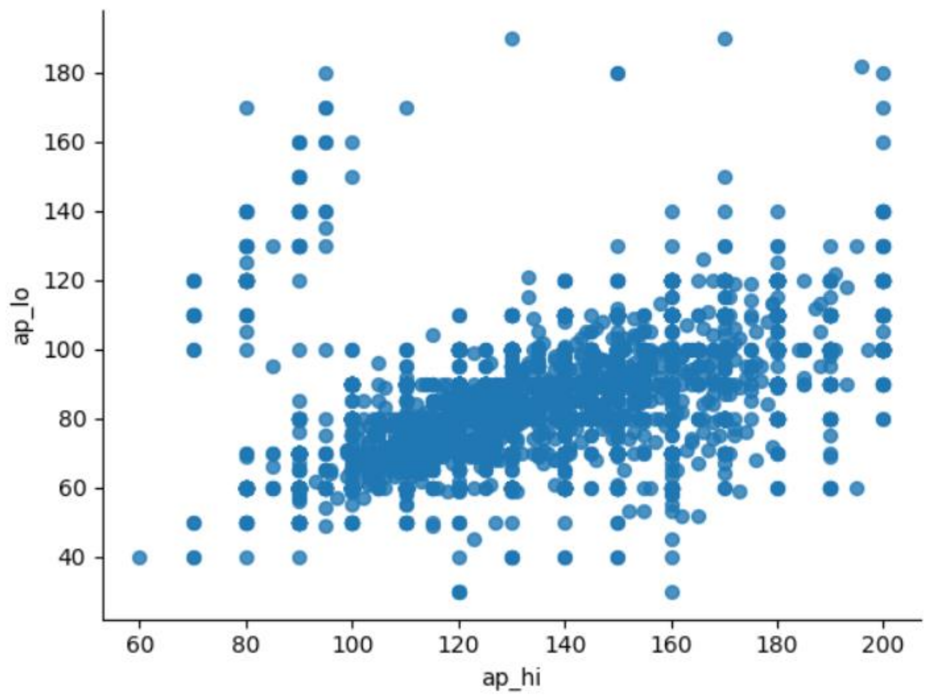


Team : omar kassar & yosr mdemagh & ala kerkeni

⇨ there is no correlation between age and weight

⇨ there is only one cluster

⇨ there are some outliers

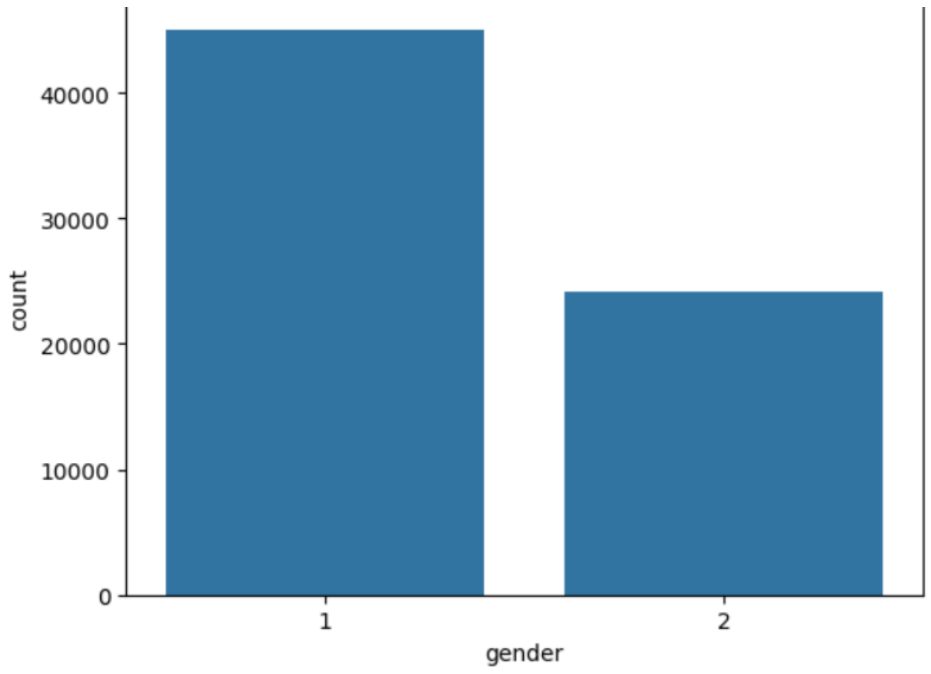


⇨ There is a moderate correlation between ap_hi and ap_lo

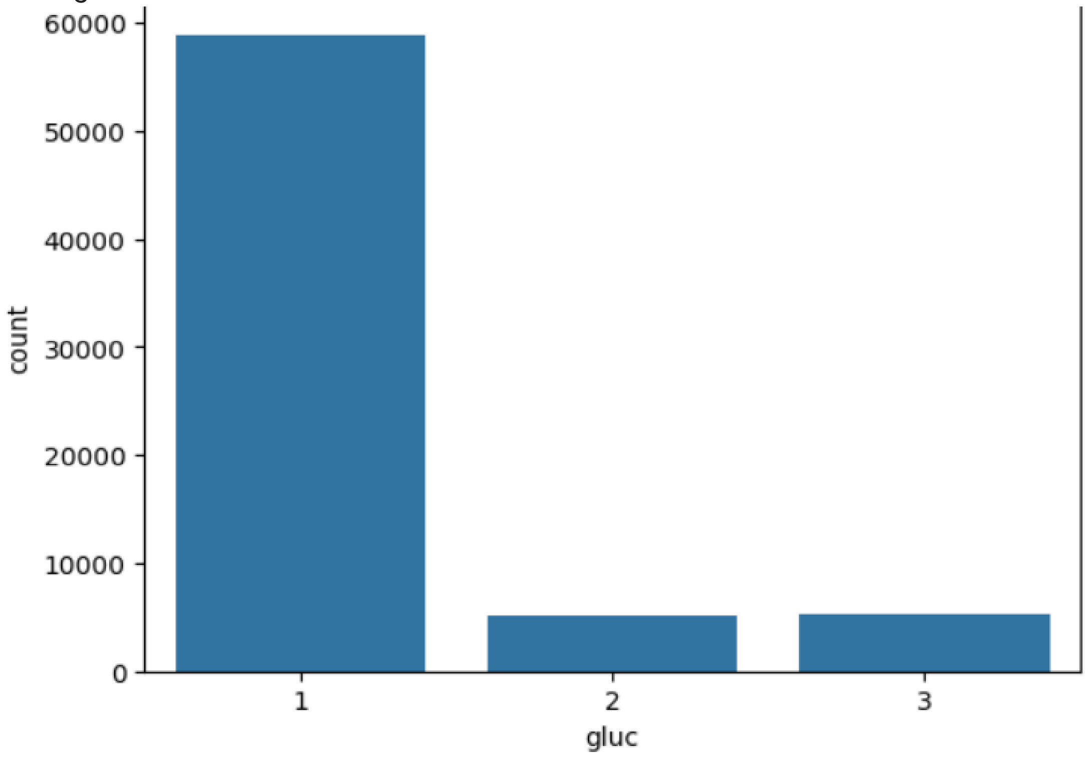
⇨ there is only one cluster

⇨ there are some outliers

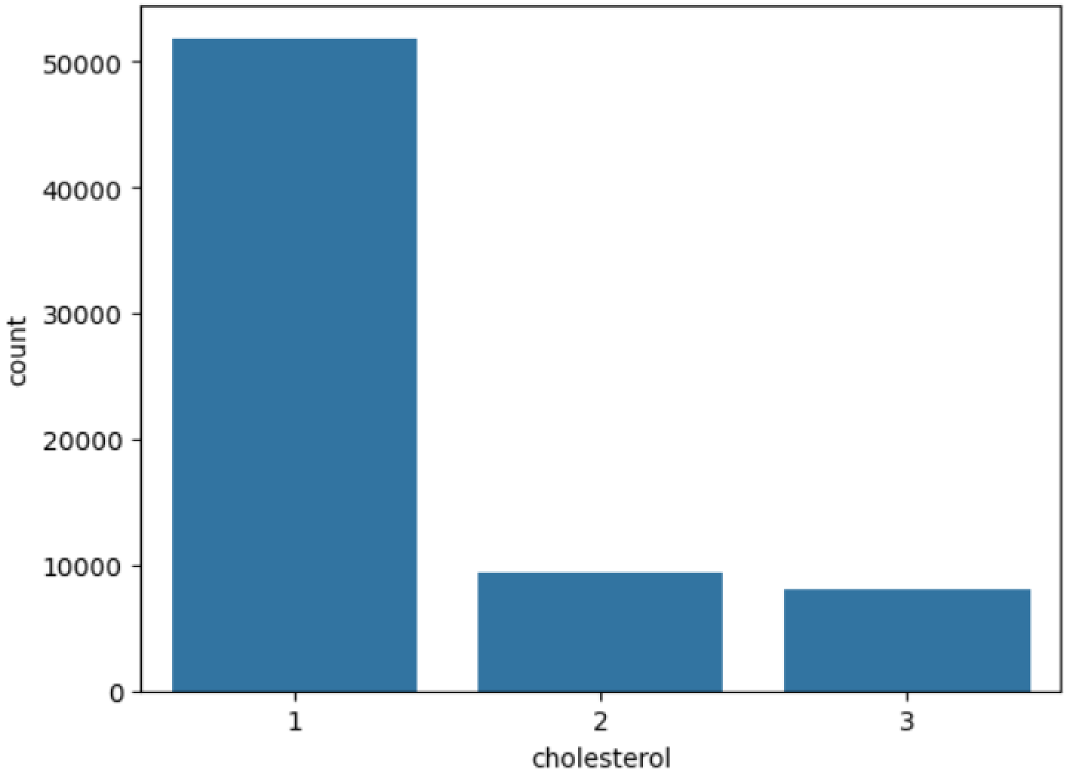
✓ **Visualizing data using bar plots:**



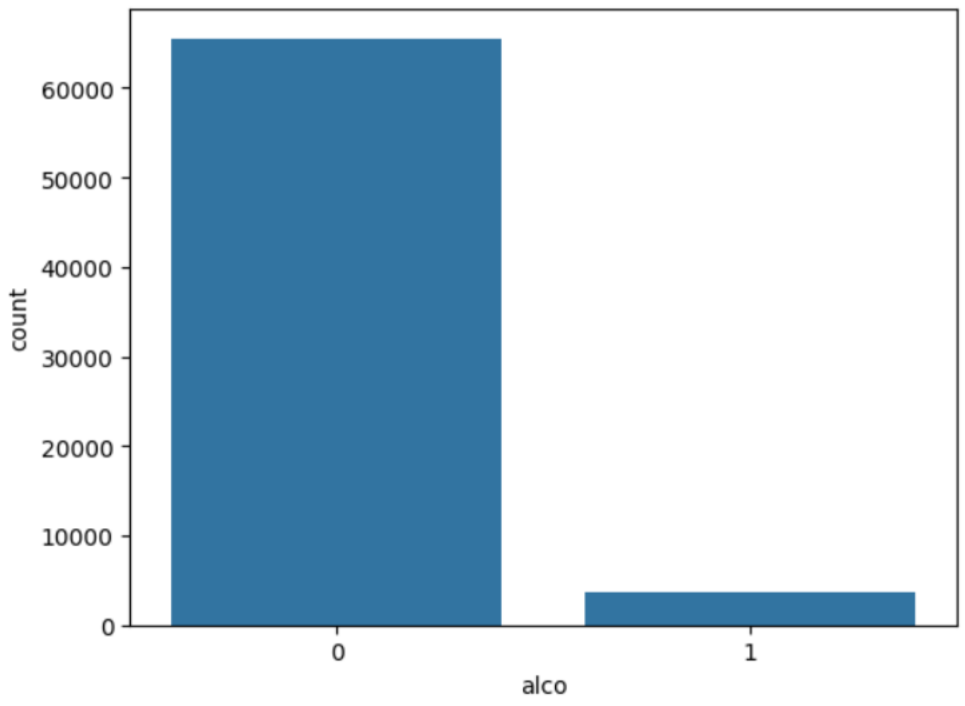
⇨ This plot shows that gender2 is represented at 50% of the frequency of gender1



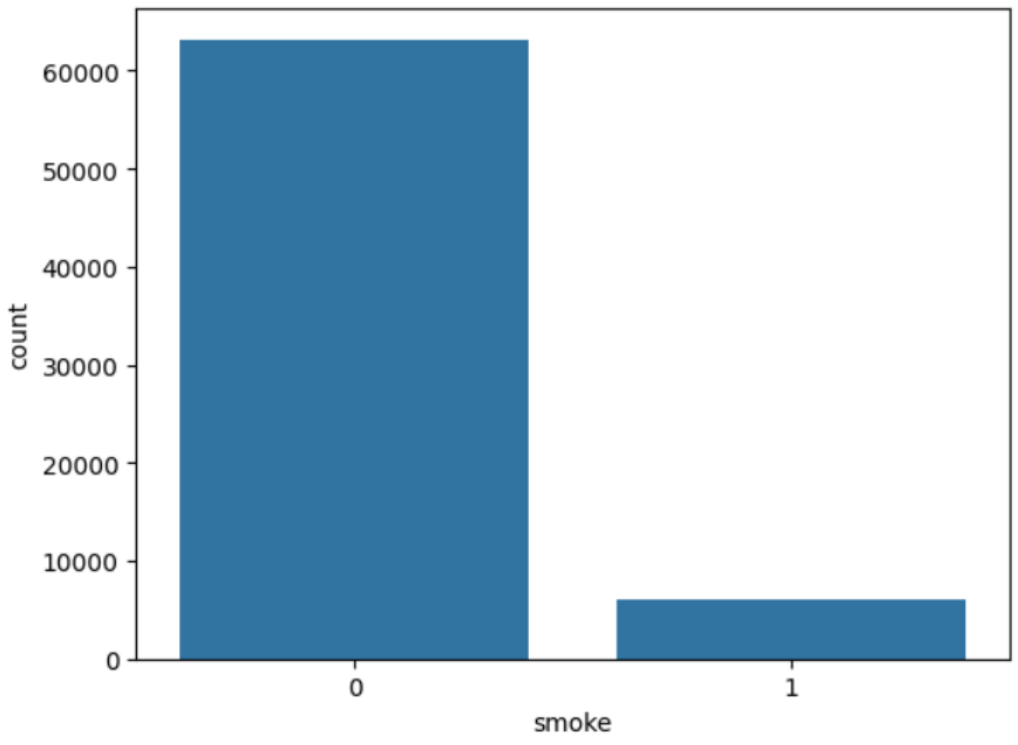
⇨ The category1 in gluc column values is the most prevalent, with other categories representing nearly one-fifth of the presence of category1.



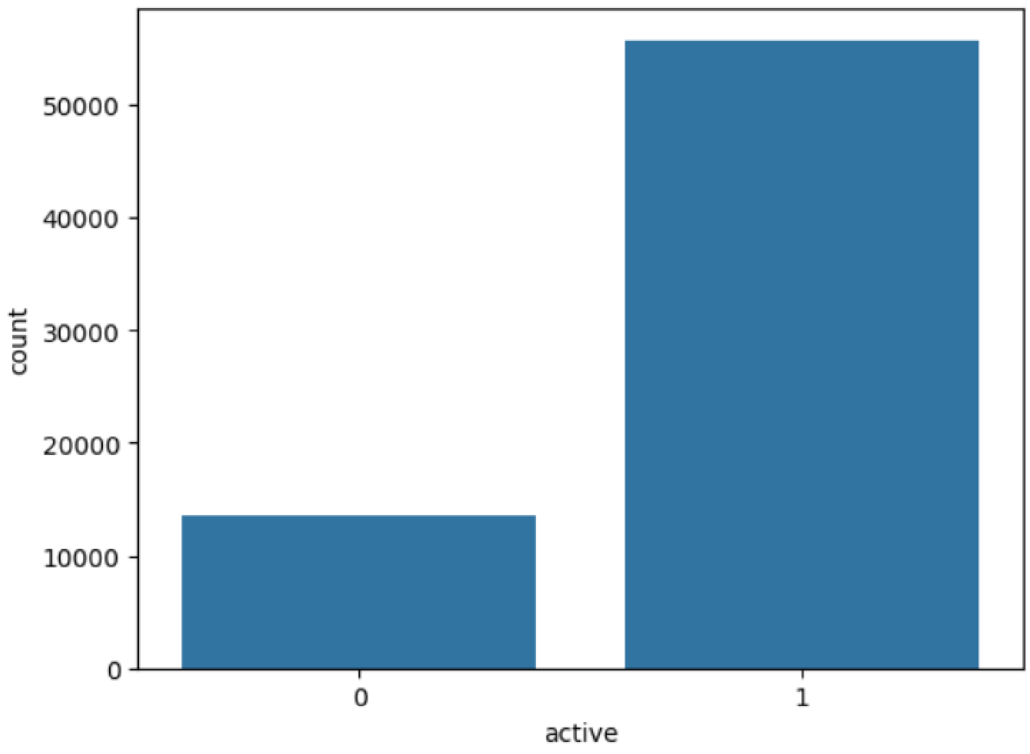
⇨ The category 1 in cholesterol column values is the most prevalent, with other categories representing nearly one-fifth of the presence of category 1.



⇨ The number of alcoholic people is nearly one-twelfth that of non-alcoholic people



⇨ The number of smokers is nearly 1/10 that of non smokers.

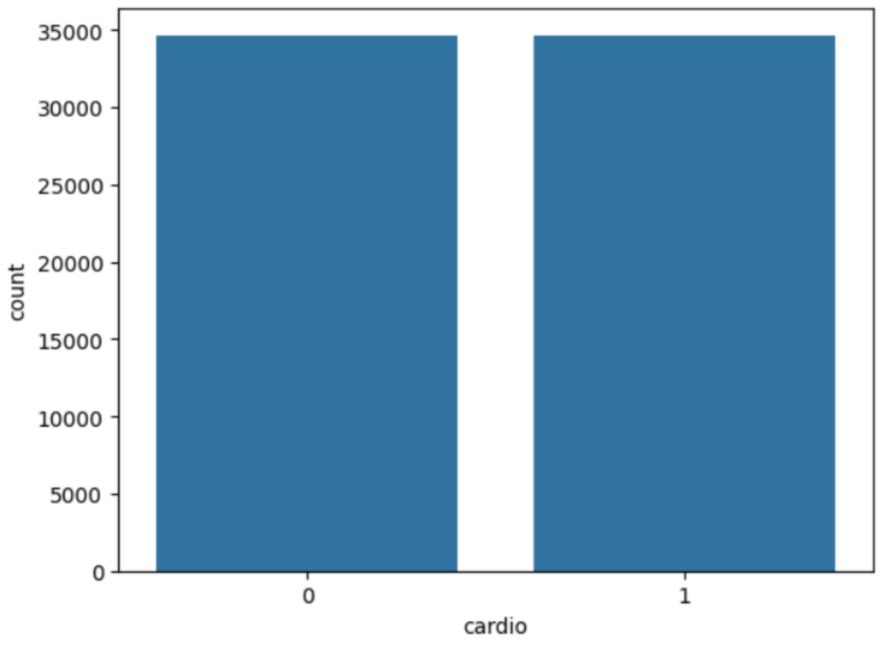


⇨ The number of inactive people is nearly one-fifth of active people

✓ **class distribution:**

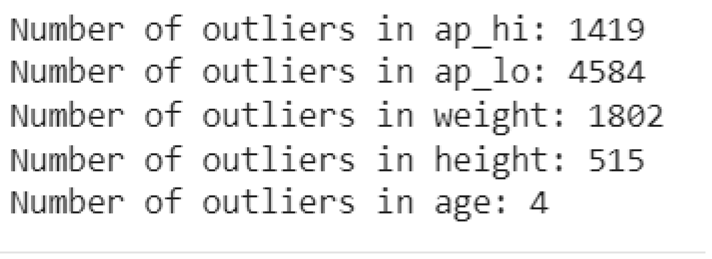
```
comptage_classes = df['cardio'].value_counts()
print((comptage_classes/df.shape[0])*100)
```

```
0    50.033922
1    49.966078
Name: cardio, dtype: float64
```



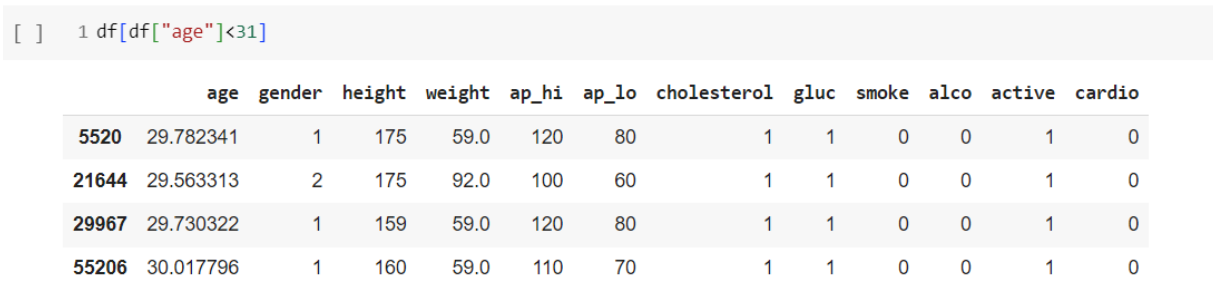
- ⇨ We have 50% of tuples with class 0 and 50% with class 1. It's notable that there is equity in the class distribution
- ⇨ we have a Balanced Representation that may have an equal impact on model's predictions
- ✓ **Dealing with outliers:**

Number of outliers :



```
Number of outliers in ap_hi: 1419
Number of outliers in ap_lo: 4584
Number of outliers in weight: 1802
Number of outliers in height: 515
Number of outliers in age: 4
```

For age :

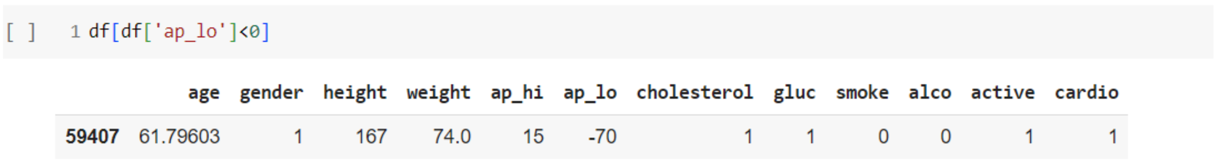


```
df[df["age"]<31]
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
5520	29.782341	1	175	59.0	120	80	1	1	0	0	1	0
21644	29.563313	2	175	92.0	100	60	1	1	0	0	1	0
29967	29.730322	1	159	59.0	120	80	1	1	0	0	1	0
55206	30.017796	1	160	59.0	110	70	1	1	0	0	1	0

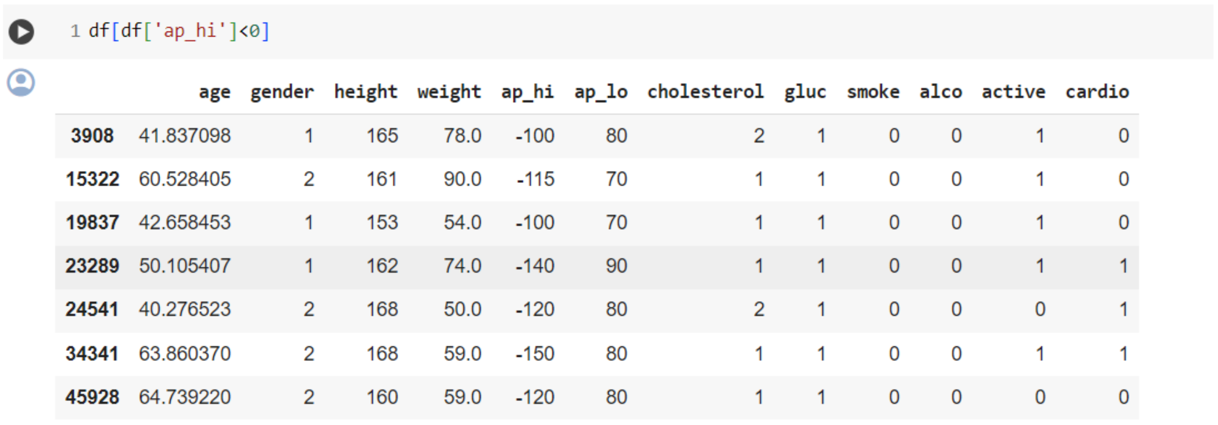
- ⇨ In the box plot, these points are considered as outliers; however, in reality, we can find people aged under 31. That's why we are going to leave them

For ap_hi and ap_lo



```
df[df['ap_lo']<0]
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
59407	61.79603	1	167	74.0	15	-70	1	1	0	0	1	1

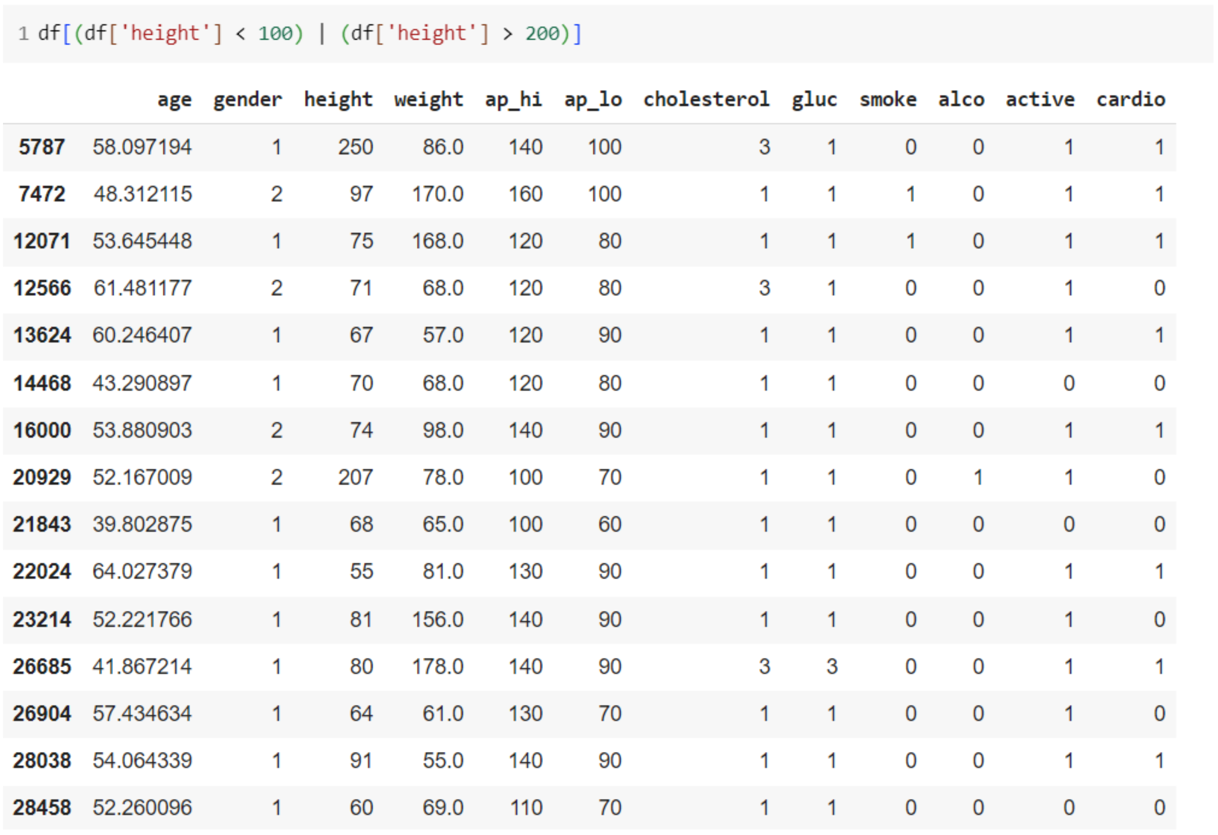


```
df[df['ap_hi']<0]
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
3908	41.837098	1	165	78.0	-100	80	2	1	0	0	1	0
15322	60.528405	2	161	90.0	-115	70	1	1	0	0	1	0
19837	42.658453	1	153	54.0	-100	70	1	1	0	0	1	0
23289	50.105407	1	162	74.0	-140	90	1	1	0	0	1	1
24541	40.276523	2	168	50.0	-120	80	2	1	0	0	0	1
34341	63.860370	2	168	59.0	-150	80	1	1	0	0	1	1
45928	64.739220	2	160	59.0	-120	80	1	1	0	0	0	0

- ⇨ There are some negative values in ap_lo and ap_hi, which is medically incorrect. However, we note that replacing them with their absolute values solves the problem and provides meaningful values. Perhaps the issue of negative values can be attributed to a data entry error during data storage
- ⇨ Medically, ap_lo can be between [30, 200], and ap_hi follows the same range. However, we found values in ap_lo and ap_hi exceeding 200, such as 16020.000000, which is an incorrect value. A possible solution is to remove these values since they will not significantly affect the dataset's dimensions; with over 69000 rows, removing 1260 rows is not critical.

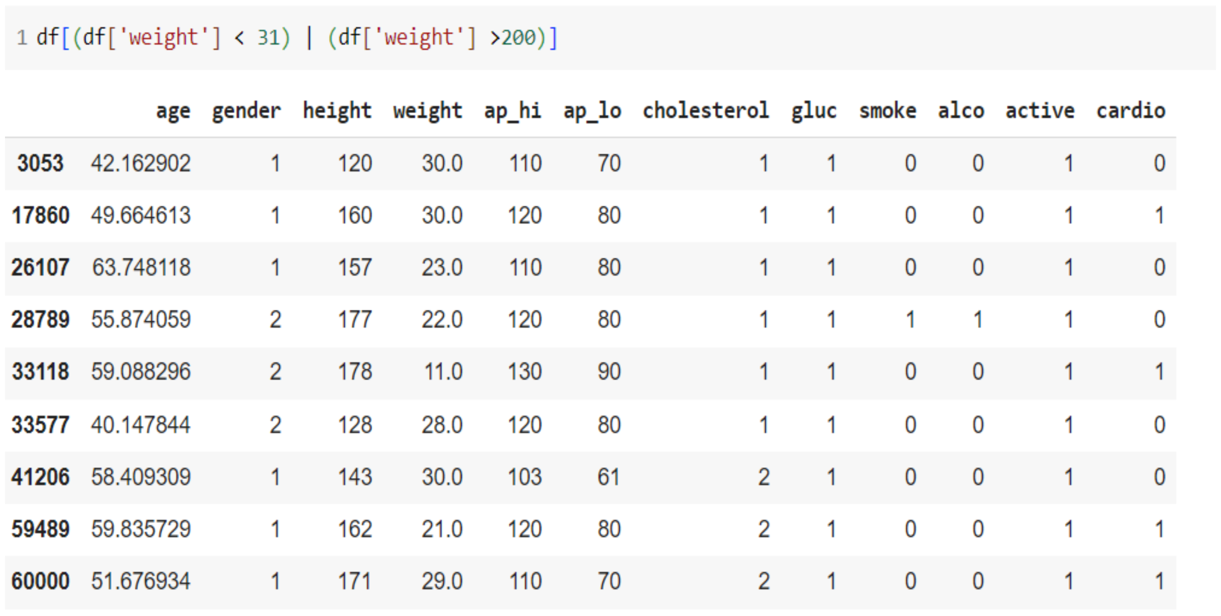
For height and weight:



```python
df[(df['height'] < 100) | (df['height'] > 200)]
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
5787	58.097194	1	250	86.0	140	100	3	1	0	0	1	1
7472	48.312115	2	97	170.0	160	100	1	1	1	0	1	1
12071	53.645448	1	75	168.0	120	80	1	1	1	0	1	1
12566	61.481177	2	71	68.0	120	80	3	1	0	0	1	0
13624	60.246407	1	67	57.0	120	90	1	1	0	0	1	1
14468	43.290897	1	70	68.0	120	80	1	1	0	0	0	0
16000	53.880903	2	74	98.0	140	90	1	1	0	0	1	1
20929	52.167009	2	207	78.0	100	70	1	1	0	1	1	0
21843	39.802875	1	68	65.0	100	60	1	1	0	0	0	0
22024	64.027379	1	55	81.0	130	90	1	1	0	0	1	1
23214	52.221766	1	81	156.0	140	90	1	1	0	0	1	0
26685	41.867214	1	80	178.0	140	90	3	3	0	0	1	1
26904	57.434634	1	64	61.0	130	70	1	1	0	0	1	0
28038	54.064339	1	91	55.0	140	90	1	1	0	0	1	1
28458	52.260096	1	60	69.0	110	70	1	1	0	0	0	0

- ⇨ people with height under 100 cm or over 200 cm can seem abnormal but in this step we will leave them as they are and in g/ in calculating the BMI we will verify whether we keep or move the row



```python
df[(df['weight'] < 31) | (df['weight'] >200)]
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
3053	42.162902	1	120	30.0	110	70	1	1	0	0	1	0
17860	49.664613	1	160	30.0	120	80	1	1	0	0	1	1
26107	63.748118	1	157	23.0	110	80	1	1	0	0	1	0
28789	55.874059	2	177	22.0	120	80	1	1	1	1	1	0
33118	59.088296	2	178	11.0	130	90	1	1	0	0	1	1
33577	40.147844	2	128	28.0	120	80	1	1	0	0	1	0
41206	58.409309	1	143	30.0	103	61	2	1	0	0	1	0
59489	59.835729	1	162	21.0	120	80	2	1	0	0	1	1
60000	51.676934	1	171	29.0	110	70	2	1	0	0	1	1

- ⇨ The boxplot indicates the presence of outliers, but upon closer examination, we discovered that individuals with a weight of less than 30 might appear abnormal. However, such individuals are typically affected by cardiovascular disease. Hence, we can retain these data points since very low weight is associated with cardiovascular disease. We will further assess this during the calculation of BMI (Body Mass Index), and if we encounter unusual values, we can consider removing them from the dataset

- ⇨ Replacing outliers in height or weight with the average, median, or mode is feasible in our case. However, given the size of our dataset (more than 69000 rows), deleting the outliers appears to be the best practice. Sacrificing less than 1% of the data to achieve optimal performance in our model is preferable to modifying outliers with other values, potentially compromising the model's accuracy.

Question 1.g:

Suggest ideas for further analysis which could be performed on the data like data transformation and data reduction. Conduct the suggested analysis and clearly explain your results.

All our columns contain numerical values, so performing data transformation isn't a good choice. Instead, we can opt for data reduction. For instance, instead of having separate columns for height and weight, we can combine them into one column named BMI (Body Mass Index), where we calculate the BMI for each person and assign them to their respective stage based on their BMI.

```python
df['BMI'] = df['weight'] / ((df['height'] / 100) ** 2)

# Print the DataFrame
df
```

instead of keeping the BMI score we can assign a BMI_stage for every person based in this rule:

Below 18.5: Underweight (1)

18.5 to 24.9: Healthy weight(2)

25 to 29.9: Overweight(3)

30 or above: Obese(4)

```python
import pandas as pd

def classify_bmi(bmi):
  """
  Classifies BMI value into weight stages.

  Args:
      bmi: A numerical value representing the Body Mass Index (BMI).

  Returns:
      An integer representing the weight stage:
          1: Underweight
          2: Healthy weight
          3: Overweight
          4: Obese
  """
  if bmi < 18.5:
    return 1  # Underweight
  elif bmi < 25:
    return 2  # Healthy weight
  elif bmi < 30:
    return 3  # Overweight
  else:
    return 4  # Obese
```

```python
# Add a new column for weight stage using a function
df['BMI_Stage'] = df['BMI'].apply(classify_bmi)
```

2.Model Engineering

Question 2.a:

Explain how you split the data into training and test sets

✓ **Data spliting :**

We split the data into two parts: 80% for training and 20% for testing

```
import pandas as pd
from sklearn.model_selection import train_test_split

#predictive variables
x=df[["age","gender","ap_hi","ap_lo","cholesterol","gluc","smoke","alco","active","BMI_Stage"]]
#target variable
y=df["cardio"]
print(x.shape,y.shape)
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=1)
```

Question 2.b:

Run the decision tree algorithm on the training data without pruning (mention the parameter setting).

Model training :

```
# Import du classificateur d'arbre de décision
from sklearn.tree import DecisionTreeClassifier

# Instanciation de l'objet du classificateur d'arbre de décision
treeClassifier = DecisionTreeClassifier()

#builduing Decision tree model
#sliding Decision treee classifier object
# Entraînement du modèle
dtree=treeClassifier.fit(xtrain, ytrain)
```

```
print("xtrain :",xtrain.shape)
print("ytrain :",ytrain.shape)
print("xtest :",xtest.shape)
print("ytest :",ytest.shape)
```

```
xtrain : (54404, 10)
ytrain : (54404,)
xtest : (13602, 10)
ytest : (13602,)
```

parameter setting

```
Criterion: gini
Max Depth: None
Min Samples Split: 2
Min Samples Leaf: 1
```

```
random_state: None
ccp_alpha: 0.0
max_depth : None
splitter : best
min_impurity_decrease : 0.0
min_weight_fraction_leaf : 0.0
```

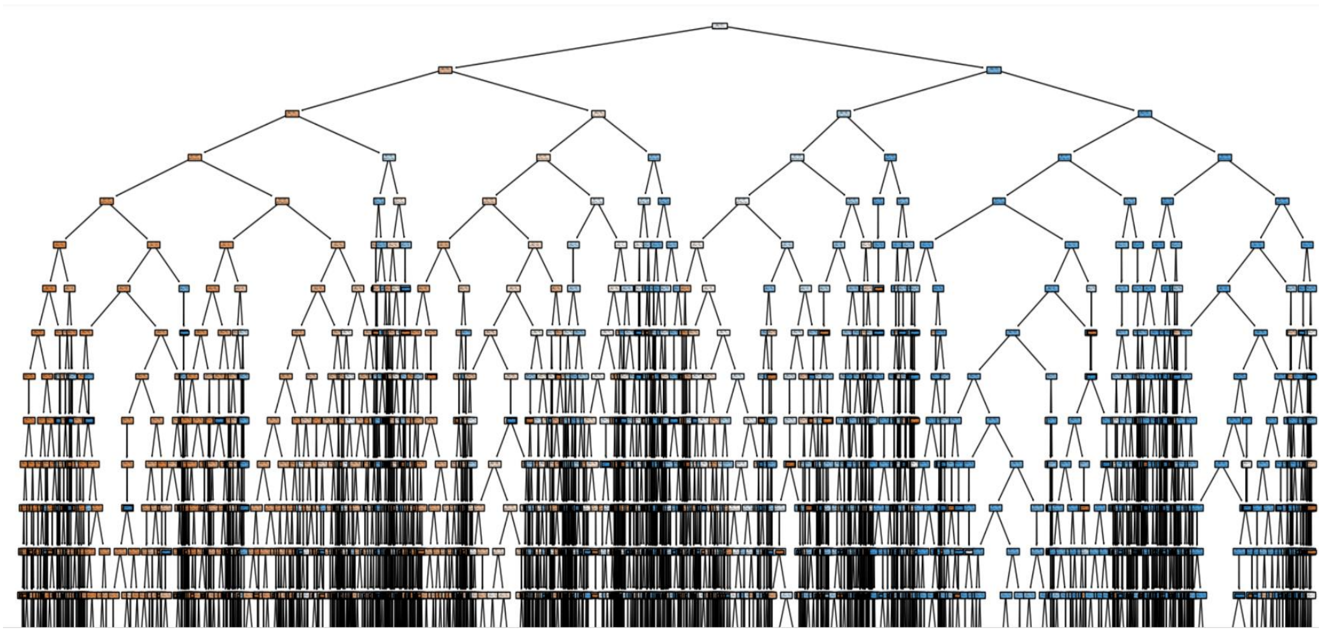
Question 2.c:

Give the graphical and textual representation of the learned decision tree

```
1 print(tree.export_text(treeClassifier))
```

```
--- feature_2 <= 129.50
    |--- feature_0 <= 54.57
    |   |--- feature_4 <= 2.50
    |   |   |--- feature_0 <= 44.08
    |   |   |   |--- feature_4 <= 1.50
    |   |   |   |   |--- feature_2 <= 114.50
    |   |   |   |   |   |--- feature_5 <= 1.50
    |   |   |   |   |   |   |--- feature_3 <= 145.00
    |   |   |   |   |   |   |   |--- feature_0 <= 43.89
    |   |   |   |   |   |   |   |   |--- feature_7 <= 0.50
    |   |   |   |   |   |   |   |   |   |--- feature_1 <= 1.50
    |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 38
    |   |   |   |   |   |   |   |   |   |--- feature_1 >  1.50
    |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 17
    |   |   |   |   |   |   |   |   |--- feature_7 >  0.50
    |   |   |   |   |   |   |   |   |   |--- feature_8 <= 0.50
    |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
    |   |   |   |   |   |   |   |   |   |--- feature_8 >  0.50
    |   |   |   |   |   |   |   |   |   |   |--- class: 0
    |   |   |   |   |   |   |   |--- feature_0 >  43.89
    |   |   |   |   |   |   |   |   |--- feature_0 <= 44.01
    |   |   |   |   |   |   |   |   |   |--- feature_2 <= 106.00
    |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 5
```

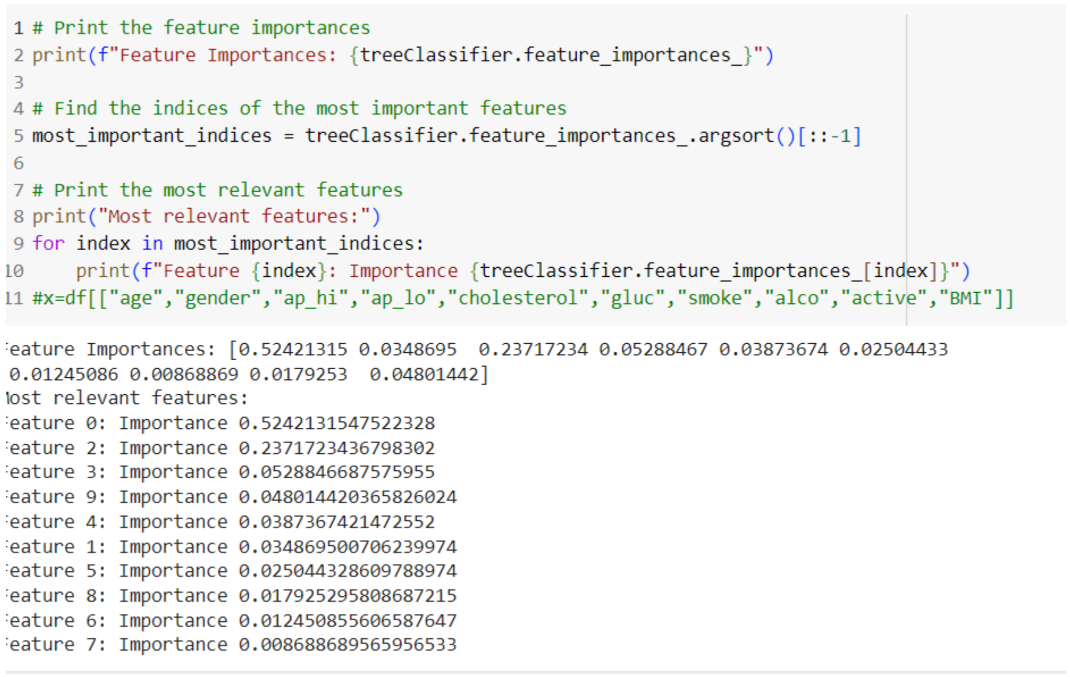
⇨ It's a part of the textual tree
Graphical tree:



Team : omar kassar & yosr mdemagh & ala kerkeni

Question 2.d:

Which features are most relevant for the classification task. Explain how the overall



```python
# Print the feature importances
print(f"Feature Importances: {treeClassifier.feature_importances_}")

# Find the indices of the most important features
most_important_indices = treeClassifier.feature_importances_.argsort()[::-1]

# Print the most relevant features
print("Most relevant features:")
for index in most_important_indices:
    print(f"Feature {index}: Importance {treeClassifier.feature_importances_[index]}")
#x=df[["age","gender","ap_hi","ap_lo","cholesterol","gluc","smoke","alco","active","BMI"]]
```

```
eature Importances: [0.52421315 0.0348695  0.23717234 0.05288467 0.03873674 0.02504433
0.01245086 0.00868869 0.0179253  0.04801442]
lost relevant features:
eature 0: Importance 0.5242131547522328
eature 2: Importance 0.2371723436798302
eature 3: Importance 0.0528846687575955
eature 9: Importance 0.048014420365826024
eature 4: Importance 0.0387367421472552
eature 1: Importance 0.034869500706239974
eature 5: Importance 0.025044328609788974
eature 8: Importance 0.017925295808687215
eature 6: Importance 0.012450855606587647
eature 7: Importance 0.008688689565956533
```

the sort of features based on their importance :

age(feature 0)

ap_hi(feature 2)

ap_lo(feature 3)

BMI(feature 9)

cholesterol(feature 4)

gender(feature 1)

gluc(feature 5)

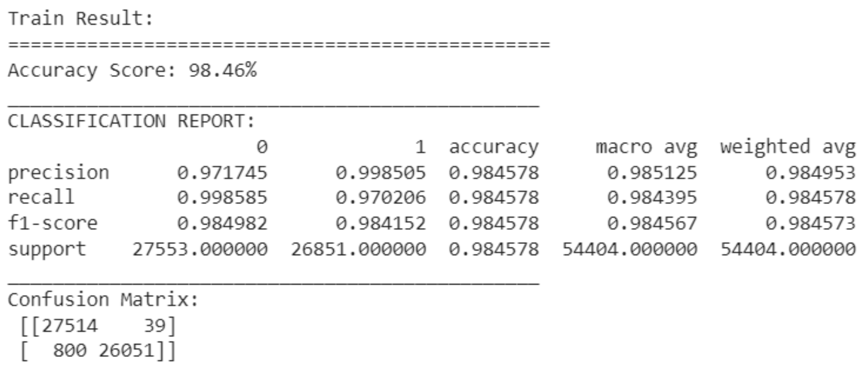
active(feature 8)

smoke(feature 6)

alco(feature 7)

Question 2.f:

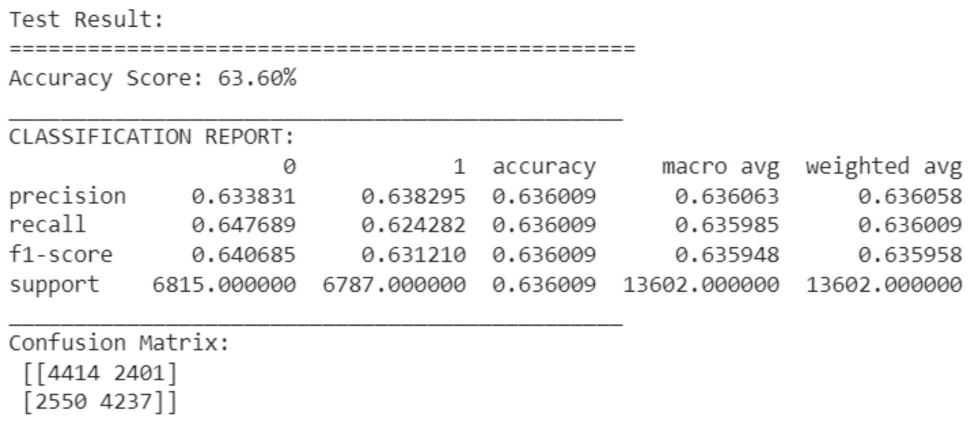
What is your learned decision tree's accuracy over the training set



```
Train Result:
================================================
Accuracy Score: 98.46%
_______________________________________________
CLASSIFICATION REPORT:
                      0             1  accuracy     macro avg  weighted avg
precision      0.971745      0.998505  0.984578      0.985125      0.984953
recall         0.998585      0.970206  0.984578      0.984395      0.984578
f1-score       0.984982      0.984152  0.984578      0.984567      0.984573
support    27553.000000  26851.000000  0.984578  54404.000000  54404.000000
_______________________________________________
Confusion Matrix:
 [[27514    39]
 [  800 26051]]
```

Question 2.g:

what is your learned decision tree's accuracy over the test set

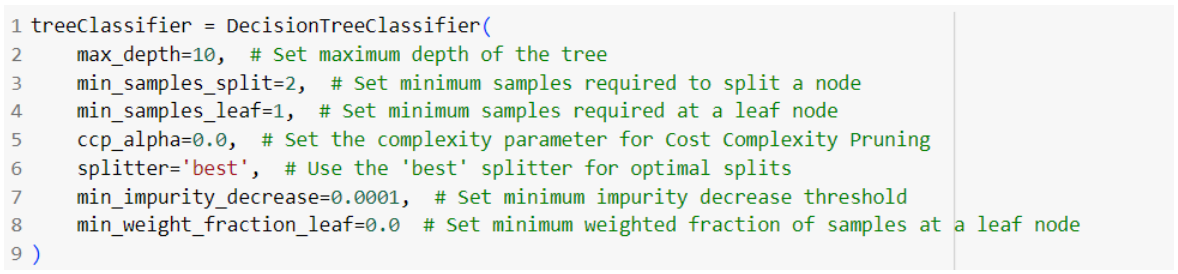


```
Test Result:
================================================
Accuracy Score: 63.60%
_______________________________________________
CLASSIFICATION REPORT:
                     0            1  accuracy     macro avg  weighted avg
precision     0.633831     0.638295  0.636009      0.636063      0.636058
recall        0.647689     0.624282  0.636009      0.635985      0.636009
f1-score      0.640685     0.631210  0.636009      0.635948      0.635958
support    6815.000000  6787.000000  0.636009  13602.000000  13602.000000
_______________________________________________
Confusion Matrix:
 [[4414 2401]
 [2550 4237]]
```

⇨ The model overfits to the training set, as the training accuracy(98,46%) is much higher than the test accuracy(63.60%)

Question 2.h:

Run the decision tree algorithm on the training data with pre-pruning (mention the parameter setting). Explain which thresholds you use and how you set them to obtain optimum results.

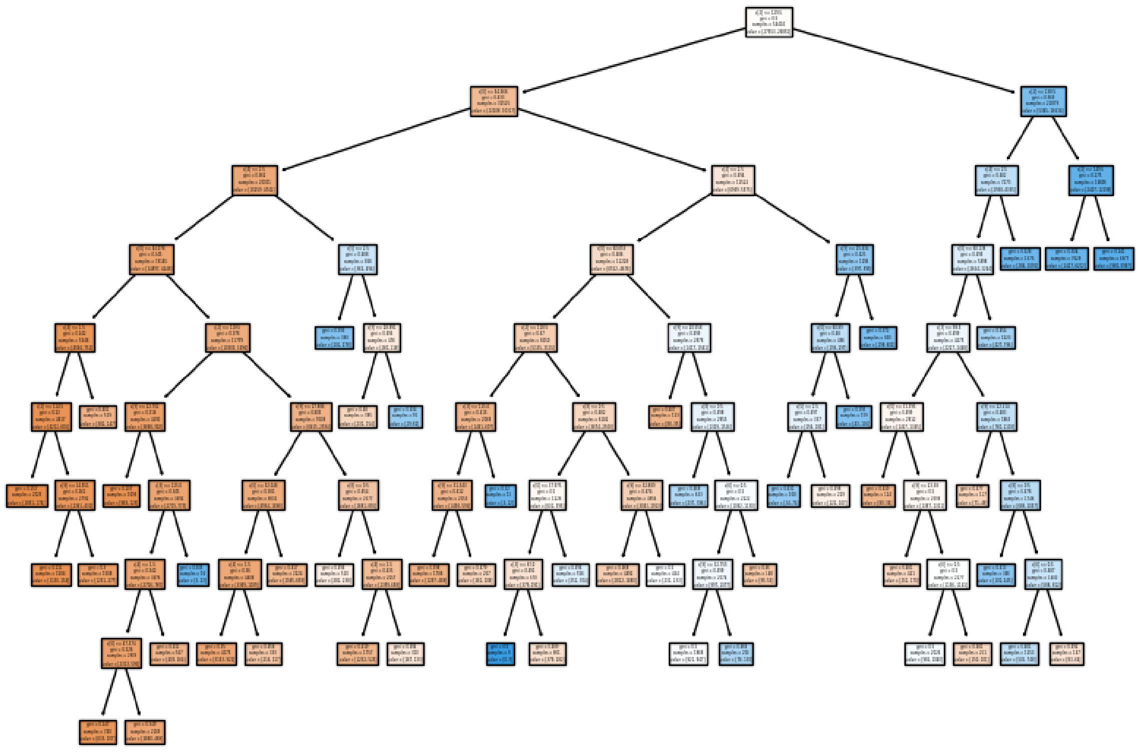


```python
treeClassifier = DecisionTreeClassifier(
    max_depth=10,  # Set maximum depth of the tree
    min_samples_split=2,  # Set minimum samples required to split a node
    min_samples_leaf=1,  # Set minimum samples required at a leaf node
    ccp_alpha=0.0,  # Set the complexity parameter for Cost Complexity Pruning
    splitter='best',  # Use the 'best' splitter for optimal splits
    min_impurity_decrease=0.0001,  # Set minimum impurity decrease threshold
    min_weight_fraction_leaf=0.0  # Set minimum weighted fraction of samples at a leaf node
)
```

Question 2.i and 2.j:

- What are the sizes of your original tree and your pruned tree?
- What are the accuracies of your unpruned and pruned trees over the training set? Over the test set?

```
Test Result:
================================================
Accuracy Score: 73.26%
_______________________________________________
CLASSIFICATION REPORT:
                     0            1  accuracy     macro avg  weighted avg
precision     0.706525     0.766588  0.732613      0.736556      0.736494
recall        0.797652     0.667305  0.732613      0.732479      0.732613
f1-score      0.749328     0.713509  0.732613      0.731419      0.731455
support    6815.000000  6787.000000  0.732613  13602.000000  13602.000000
_______________________________________________
Confusion Matrix:
 [[5436 1379]
 [2258 4529]]
```



Question 1.k:

- Implement a post-pruning procedure for your learned decision tree (without pruning). You may choose reduced error pruning, rule post-pruning, or any other method you choose so long as you describe it precisely (Note: if you use rule post-pruning, make up a reasonable definition of "size"). Explain if you need a validation set for postpruning. If yes, explain how you split your data into training/validation/test sets. (Note: some methods make post-pruning on training set).

Minimal Cost-Complexity Pruning

$R_\alpha(T)=R(T)+\alpha|T|$

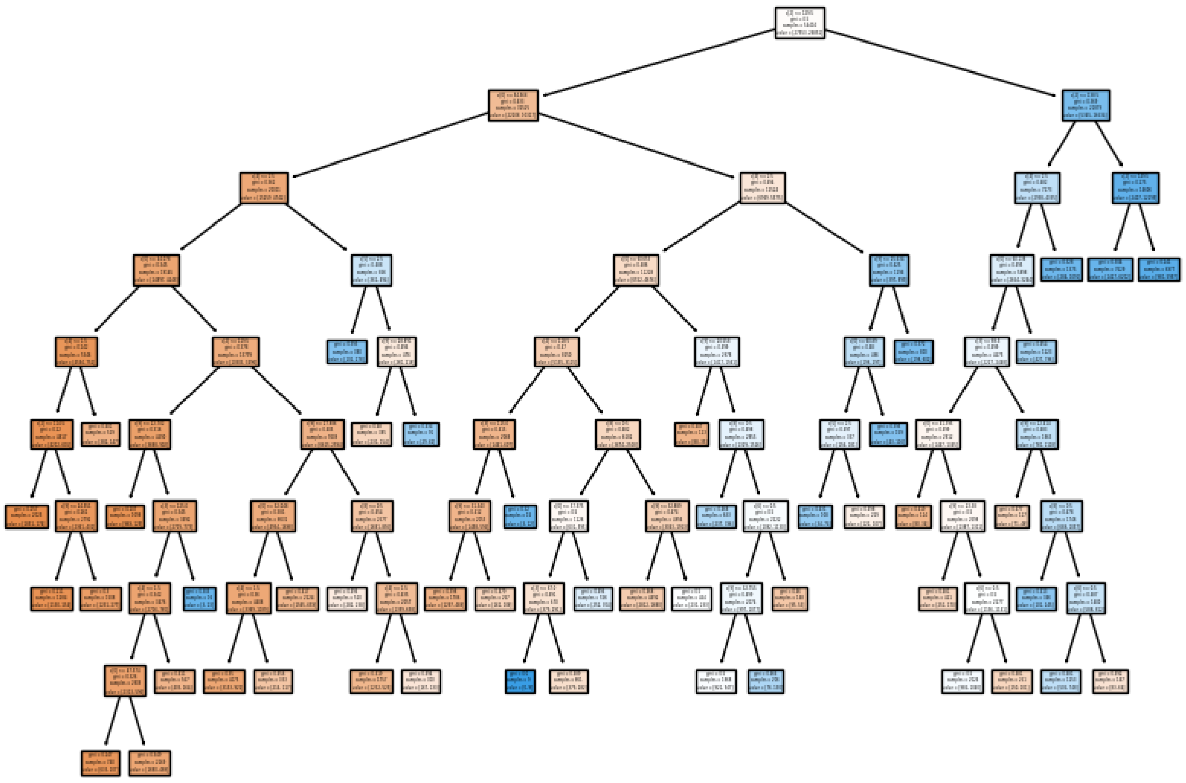
- $R(T)$is the misclassification rate of the tree T. R(T)=Number of Incorrect Predictions/Total Number of Predictions
- $|T|$is the number of terminal nodes (leaves) in the tree T.
- $\alpha$is a non-negative real number known as the complexity parameter

The goal in minimal cost-complexity pruning is to find the subtree $T(\alpha)$ that minimizes $R_\alpha(T)$ for a given α

Question 2.l and 2.m:

- **What are the sizes of your original tree and your pruned tree?**
- **What are the accuracies of your unpruned and pruned trees over the training set? Over the test set? Over the validation set.**

Prepruned tree:



- Accuracy:
  72.91% (on the test set)

- 73.23% (on validation set)
  73.72% (on the training set)

- Number of nodes :
- post pruned tree: 97
- original tree: 32129

Question 1.n , 2.o and 2.p:

the accuracy of the pre pruned tree 73.26% is slightly bigger than the accuracy of the post pruned tree 72.91%.
The original tree overfits to the training set, as the training accuracy(98.46%) is much higher than the test accuracy(63.25%).

We recommend using the post-pruned tree for classifying future data. Although it has a slightly lower accuracy on the training set compared to the pre-pruned tree, its pruning process likely leads to better generalization and less overfitting.

Combining pre-pruning and post-pruning can help create a more robust decision tree model by controlling its size during growth and further refining its structure to improve generalization