**Predict Smoking status and Finding the Accuracy of**

**GPT Bantwal**

**the Model**

**Introduction :**

In this project we are predicting the status of smoker using different models such as DecisionTreeClassifier and RandomForestClassifier.With this we can identify the characteristics of smokers. Also finding the accuracy of the Model and Comparing different Models.

About Dataset

##### ***This dataset is a collection of basic health biological signal data.***

* The goal is to determine the presence or absence of smoking through bio-signals.
* The dataset is divided into type.

## **Data Exploration & Analysis**

**#Importing the basic libraries for analysis**

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

plt.style.use("ggplot ")

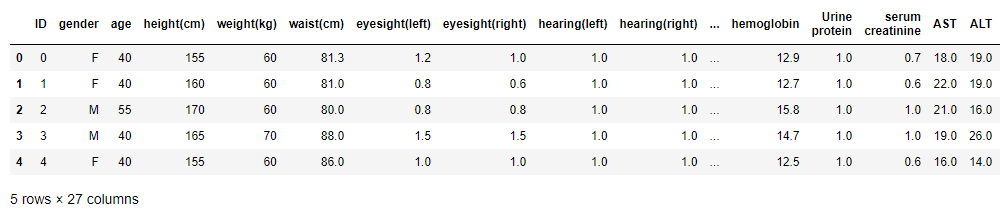
%matplotlib inline

import plotly.graph\_objects as go

import plotly.express as p

df =pd.read\_csv("C:\\Users\\GPT-BANTWAL\\AI\\smoking.csv")

df.head()



**# looking the shape DataSet**

df.shape

(55692, 27)

**#Checking the dtypes of all the columns**

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 55692 entries, 0 to 55691

Data columns (total 27 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ID 55692 non-null int64

1 gender 55692 non-null object

2 age 55692 non-null int64

3 height(cm) 55692 non-null int64

4 weight(kg) 55692 non-null int64

5 waist(cm) 55692 non-null float64

6 eyesight(left) 55692 non-null float64

7 eyesight(right) 55692 non-null float64

8 hearing(left) 55692 non-null float64

9 hearing(right) 55692 non-null float64

10 systolic 55692 non-null float64

11 relaxation 55692 non-null float64

12 fasting blood sugar 55692 non-null float64

13 Cholesterol 55692 non-null float64

14 triglyceride 55692 non-null float64

15 HDL 55692 non-null float64

16 LDL 55692 non-null float64

17 hemoglobin 55692 non-null float64

18 Urine protein 55692 non-null float64

19 serum creatinine 55692 non-null float64

20 AST 55692 non-null float64

21 ALT 55692 non-null float64

22 Gtp 55692 non-null float64

23 oral 55692 non-null object

24 dental caries 55692 non-null int64

25 tartar 55692 non-null object

26 smoking 55692 non-null int64

dtypes: float64(18), int64(6), object(3)

memory usage: 11.5+ MB

**#Checking the dtypes of all the columns**

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 55692 entries, 0 to 55691

Data columns (total 27 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ID 55692 non-null int64

1 gender 55692 non-null object

2 age 55692 non-null int64

3 height(cm) 55692 non-null int64

4 weight(kg) 55692 non-null int64

5 waist(cm) 55692 non-null float64

6 eyesight(left) 55692 non-null float64

7 eyesight(right) 55692 non-null float64

8 hearing(left) 55692 non-null float64

9 hearing(right) 55692 non-null float64

10 systolic 55692 non-null float64

11 relaxation 55692 non-null float64

12 fasting blood sugar 55692 non-null float64

13 Cholesterol 55692 non-null float64

14 triglyceride 55692 non-null float64

15 HDL 55692 non-null float64

16 LDL 55692 non-null float64

17 hemoglobin 55692 non-null float64

18 Urine protein 55692 non-null float64

19 serum creatinine 55692 non-null float64

20 AST 55692 non-null float64

21 ALT 55692 non-null float64

22 Gtp 55692 non-null float64

23 oral 55692 non-null object

24 dental caries 55692 non-null int64

25 tartar 55692 non-null object

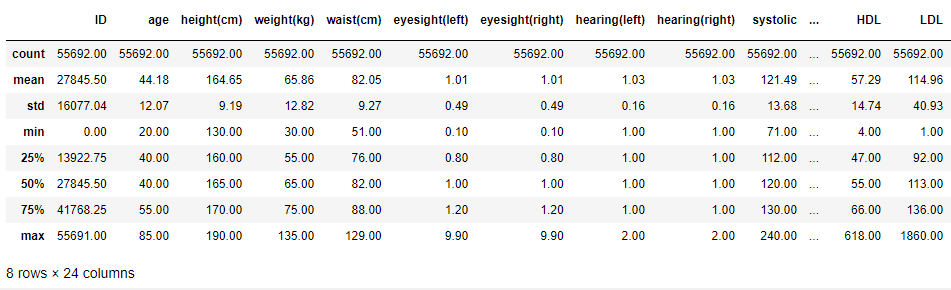
26 smoking 55692 non-null int64

dtypes: float64(18), int64(6), object(3)

memory usage: 11.5+ MB

**# look describe data set**

df.describe().round(2)



**# check unique value**

df.nunique().sort\_values()

oral 1

smoking 2

gender 2

dental caries 2

hearing(left) 2

hearing(right) 2

tartar 2

Urine protein 6

height(cm) 13

age 14

eyesight(right) 17

eyesight(left) 19

weight(kg) 22

serum creatinine 38

relaxation 95

HDL 126

systolic 130

hemoglobin 145

AST 219

ALT 245

fasting blood sugar 276

Cholesterol 286

LDL 289

triglyceride 390

Gtp 488

waist(cm) 566

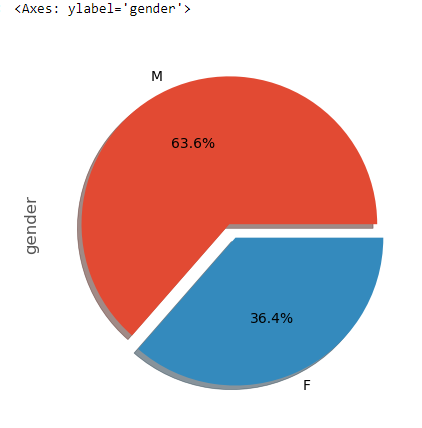
ID 55692

dtype: int64

## **Some of Visualisations**

**# how much percentage Gender in the dataset**

df['gender'].value\_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',shadow=True)



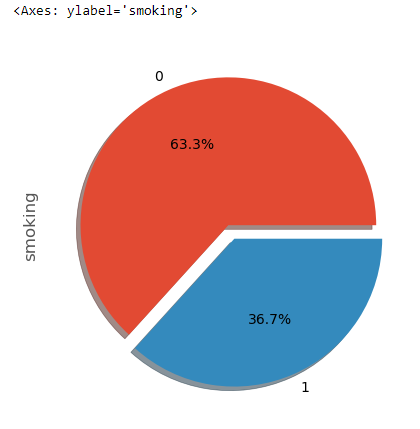
### The percentage gender in the dataset:

Female = 36.4%

Male =63.6 %

**# how much percentage smoking in the dataset**

df['smoking'].value\_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',shadow=True)



### The percentage gender in the dataset:

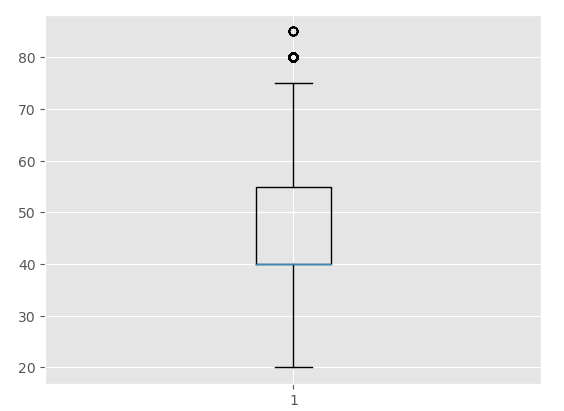
Non-smoking = 63.3%

Smoking =36.7 %

**# boxplot for show describe age**

plt.boxplot(df["age"])

plt.show()



### Describe age :

mean = 44

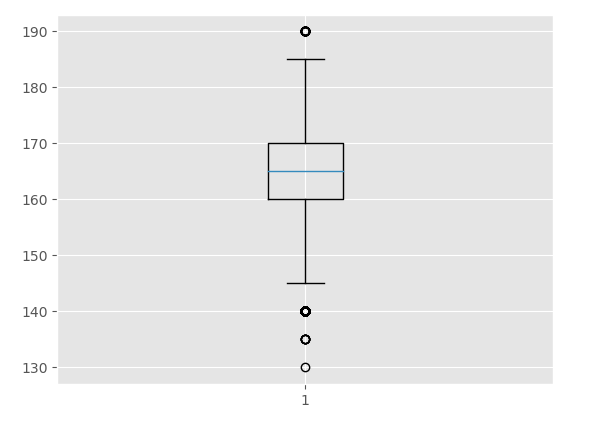
min = 20

max = 85

**# boxplot for show describe height**

plt.boxplot(df["height(cm)"])

plt.show()

****

### Describe height- cm :

mean= 164

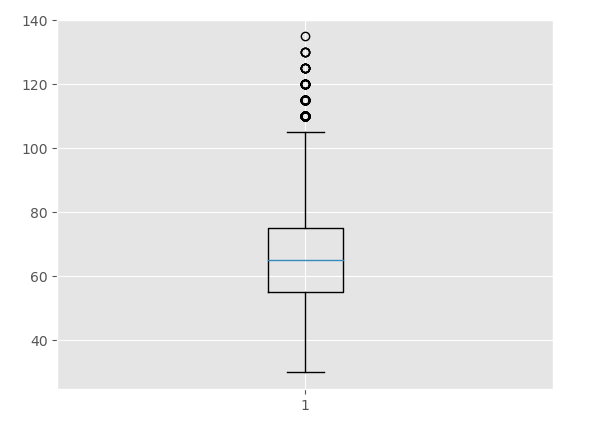
min = 130

max = 190

**# boxplot for show describe weight**

plt.boxplot(df["weight(kg)"])

plt.show()



### Describe weight(kg) :

mean= 65

min = 30

max = 135

**#make groupby to show the average age smoking**

ag=df.groupby("smoking")["age"].mean()

ag

smoking

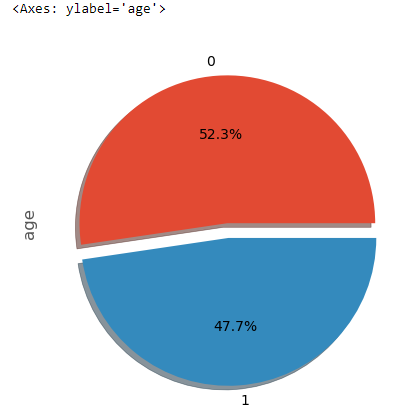
0 45.677981

1 41.607431

Name: age, dtype: float64

**# graph average age smoking**

ag.plot(kind="pie",explode=[0,0.1],autopct='%1.1f%%',shadow=True)



### The average age smoking

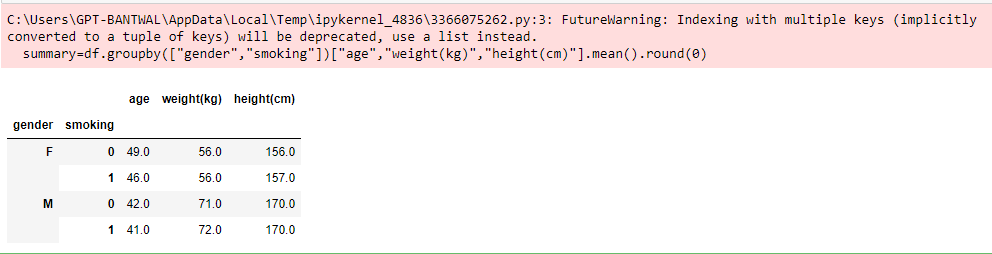
age 45 = non- smoking

age 41 = smoking

**# group by for show the average age , weight and height by the gender**

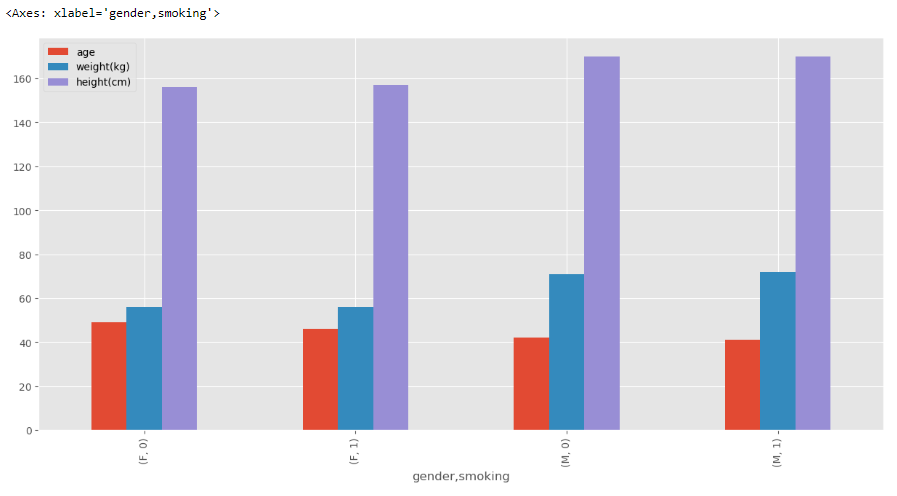
summary=df.groupby(["gender","smoking"])["age","weight(kg)","height(cm)"].mean().round(0)

summary



**# graph the group by**

summary.plot(kind="bar",figsize=(15,7))



### Analysis Results

#### **From my point view , I see the important columns :**

Gender , smoking , age , weight(kg) , height(cm)

#### **After make some analysis , visual graph and explore the data set , I see some results .**

##### ***The shape DataSet***

Rows= 55692 , Columns = 27

##### ***The percentage gender in the dataset:***

Female = 36.4%

Male =63.6 %

##### ***The percentage gender in the dataset:***

Non-smoking = 63.3%

Smoking =36.7 %

##### ***Describe age :***

mean = 44

min = 20

max = 85

##### ***Describe height- cm :***

mean= 164

min = 130

max = 190

##### ***Describe weight(kg) :***

mean= 65

min = 30

max = 135

##### ***The average age smoking***

age 45 = non- smoking

age 41 = smoking

##### ***The average data***

###### ***Female***

somking avg /> age= 46 ,weght =56 kg ,height 157 cm

non-somking ave / >age= 49 ,weght =56 kg ,height 165 cm

###### ***Male***

somking avg /> age= 41 ,weght =72 kg ,height 170 cm

non-somking ave / >age= 42 ,weght =71 kg ,height 170 cm

### Building a Machine Learning Model / classification score Volume

X\_train = pd.read\_csv('C:\\Users\\GPT-BANTWAL\\AI\\x\_train.csv\\x\_train.csv')

X\_test = pd.read\_csv('C:\\Users\\GPT-BANTWAL\\AI\\x\_test.csv\\x\_test.csv')

y\_train = pd.read\_csv('C:\\Users\\GPT-BANTWAL\\AI\\y\_train.csv')

y\_test = pd.read\_csv('C:\\Users\\GPT-BANTWAL\\AI\\y\_test.csv')

l = [X\_train,X\_test,y\_train,y\_test]

for i in l:

i.info()

print('====================================\n\n\n')

del X\_train["oral"]

del X\_test["oral"]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 44553 entries, 0 to 44552

Data columns (total 26 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ID 44553 non-null int64

1 gender 44553 non-null object

2 age 44553 non-null int64

3 height(cm) 44553 non-null int64

4 weight(kg) 44553 non-null int64

5 waist(cm) 44553 non-null float64

6 eyesight(left) 44553 non-null float64

7 eyesight(right) 44553 non-null float64

8 hearing(left) 44553 non-null float64

9 hearing(right) 44553 non-null float64

10 systolic 44553 non-null float64

11 relaxation 44553 non-null float64

12 fasting blood sugar 44553 non-null float64

13 Cholesterol 44553 non-null float64

14 triglyceride 44553 non-null float64

15 HDL 44553 non-null float64

16 LDL 44553 non-null float64

17 hemoglobin 44553 non-null float64

18 Urine protein 44553 non-null float64

19 serum creatinine 44553 non-null float64

20 AST 44553 non-null float64

21 ALT 44553 non-null float64

22 Gtp 44553 non-null float64

23 oral 44553 non-null object

24 dental caries 44553 non-null float64

25 tartar 44553 non-null object

dtypes: float64(19), int64(4), object(3)

memory usage: 8.8+ MB

====================================

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 11139 entries, 0 to 11138

Data columns (total 26 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ID 11139 non-null int64

1 gender 11139 non-null object

2 age 11139 non-null int64

3 height(cm) 11139 non-null int64

4 weight(kg) 11139 non-null int64

5 waist(cm) 11139 non-null float64

6 eyesight(left) 11139 non-null float64

7 eyesight(right) 11139 non-null float64

8 hearing(left) 11139 non-null float64

9 hearing(right) 11139 non-null float64

10 systolic 11139 non-null float64

11 relaxation 11139 non-null float64

12 fasting blood sugar 11139 non-null float64

13 Cholesterol 11139 non-null float64

14 triglyceride 11139 non-null float64

15 HDL 11139 non-null float64

16 LDL 11139 non-null float64

17 hemoglobin 11139 non-null float64

18 Urine protein 11139 non-null float64

19 serum creatinine 11139 non-null float64

20 AST 11139 non-null float64

21 ALT 11139 non-null float64

22 Gtp 11139 non-null float64

23 oral 11139 non-null object

24 dental caries 11139 non-null float64

25 tartar 11139 non-null object

dtypes: float64(19), int64(4), object(3)

memory usage: 2.2+ MB

====================================

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 44553 entries, 0 to 44552

Data columns (total 2 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ID 44553 non-null int64

1 smoking 44553 non-null int64

dtypes: int64(2)

memory usage: 696.3 KB

====================================

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 11139 entries, 0 to 11138

Data columns (total 2 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ID 11139 non-null int64

1 smoking 11139 non-null int64

dtypes: int64(2)

memory usage: 174.2 KB

====================================

l3 = [X\_train,X\_test]

from sklearn.preprocessing import OneHotEncoder

from sklearn.preprocessing import OrdinalEncoder

ONE = OneHotEncoder(handle\_unknown='ignore')

def oneHot(df,a):

cat\_encoder = OneHotEncoder()

ec\_cat=cat\_encoder.fit\_transform(df[[a]])

return ec\_cat.toarray()

X\_train['gender'] = oneHot(X\_train,'gender')

X\_test['gender'] = oneHot(X\_test,'gender')

from sklearn.preprocessing import OrdinalEncoder

ordinal\_encoder = OrdinalEncoder(categories = [['N','Y']])

X\_train["tartar"] = ordinal\_encoder.fit\_transform(X\_train[["tartar"]])

X\_test["tartar"] = ordinal\_encoder.fit\_transform(X\_test[["tartar"]])

**#Importing the basic librarires for building model - classification**

from sklearn.metrics import accuracy\_score,r2\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neural\_network import MLPClassifier

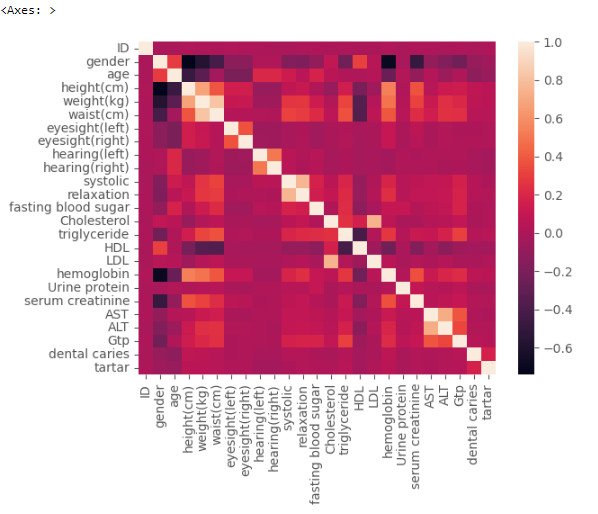
from sklearn.svm import SVC

from xgboost import XGBClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn.inspection import permutation\_importance

sns.heatmap(X\_train.corr())



y\_train = y\_train['smoking']

y\_test = y\_test['smoking']

### DecisionTreeClassifier

DTC = DecisionTreeClassifier()

DTC.fit(X\_train, y\_train)

y\_pred = DTC.predict(X\_test)

print("Score the X-train with Y-train is : ", DTC.score(X\_train,y\_train))

print("Score the X-test with Y-test is : ", DTC.score(X\_test,y\_test))

print("Accuracy Score :",accuracy\_score(y\_test,y\_pred)\*100)

Score the X-train with Y-train is : 1.0

Score the X-test with Y-test is : 0.9366190860939043

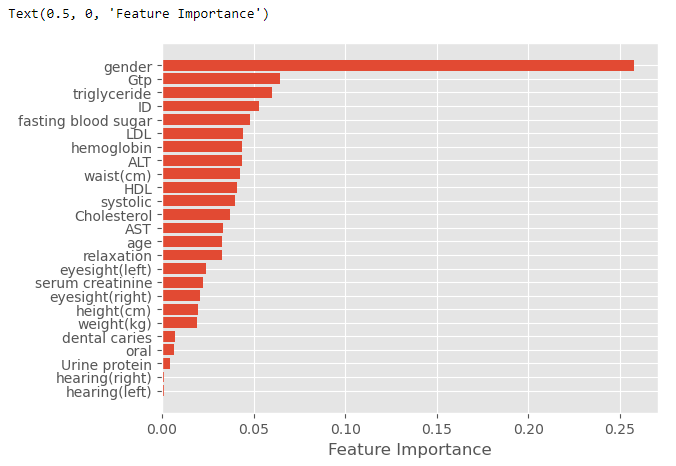
Accuracy Score : 93.66190860939044

**#feature\_importances**

sort = DTC.feature\_importances\_.argsort()

plt.barh(df.columns[sort], DTC.feature\_importances\_[sort])

plt.xlabel("Feature Importance")



### RandomForestClassifier

rf = RandomForestClassifier(n\_estimators=100)

rf.fit(X\_train, y\_train)

ypred = rf.predict(X\_test)

print("Score the X-train with Y-train is : ", rf.score(X\_train,y\_train))

print("Score the X-test with Y-test is : ", rf.score(X\_test,y\_test))

print("Accuracy Score :",accuracy\_score(y\_test,ypred)\*100)

Score the X-train with Y-train is : 0.9999775548223464

Score the X-test with Y-test is : 0.9918305054313673

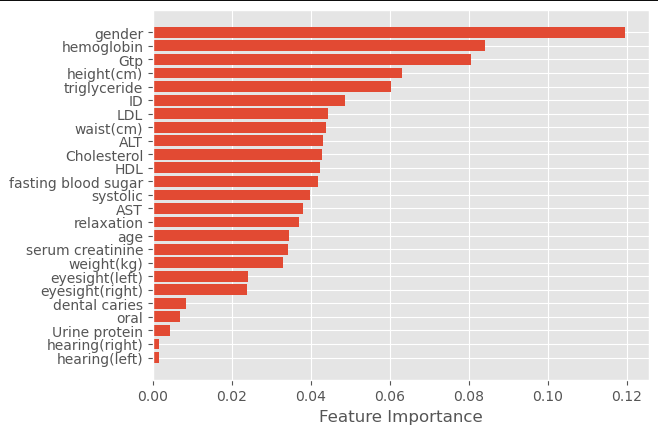
Accuracy Score : 99.18305054313673

**#feature\_importances**

sort = rf.feature\_importances\_.argsort()

plt.barh(df.columns[sort], rf.feature\_importances\_[sort])

plt.xlabel("Feature Importance")



### Model Selection Results

Decision Tree Classifier = 93.59906634347787%

Random Forest Classifier = 99.23691534249035%