Heart disease prediction

JosueARz

Heart disease prediction

The dataset used for this exercise is publicly available on the Kaggle website and comes from a cardiovascular study of residents of the city of Framingham, Massachusetts.

The objective of the classification will be to predict whether the patient has a 10-year risk of future coronary artery disease (CHD).

Import libraries

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
%matplotlib inline
import statsmodels.api as sm
from statsmodels.tools import add_constant
import scipy.stats as st
from sklearn.metrics import confusion_matrix
```

[2]:
```
sns.set(style = 'darkgrid')
sns.set_palette('deep')
```

Read data

[4]:
```
data=pd.read_csv('framingham.csv', header=0)
```

The data set is composed of 4238 observations and 16 variables, which are:

Variables :

Each attribute is a potential risk factor. There are both demographic,behavioural and medical risk factors.

Demographic:

sex: male or female (binary: “1”, means “male”, “0” means “female”)

age: age of the patient

Behaviors

currentSmoker: whether or not the patient is a current smoker

cigsPerDay: the number of cigarettes that the person smoked on average in one day.

Medical history:

BPMeds: whether or not the patient was on blood pressure medication

prevalentStroke: whether or not the patient had previously had a stroke

prevalentHyp: whether or not the patient was hypertensive

diabetes: whether or not the patient had diabetes

Vital signs :

totChol: total cholesterol level

sysBP: systolic blood pressure

diaBP: diastolic blood pressure

BMI: Body Mass Index

heartRate: heart rate

glucose: glucose level

Predict variable:

10 year risk of coronary heart disease CHD (binary: “1”, means “Yes”, “0” means “No”)

Data Exploration

[5]:
```
data.head()
```

[5]:
```
   male  age  education  currentSmoker  cigsPerDay  BPMeds ␣
↪prevalentStroke  \
0     1   39        4.0              0         0.0     0.0                ␣
 ↪  0
1     0   46        2.0              0         0.0     0.0                ␣
 ↪  0
2     1   48        1.0              1        20.0     0.0                ␣
 ↪  0
3     0   61        3.0              1        30.0     0.0                ␣
 ↪  0
4     0   46        3.0              1        23.0     0.0                ␣
 ↪  0

   prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  heartRate ␣
↪glucose  \
0             0         0    195.0  106.0   70.0  26.97       80.0   ␣
↪77.0
1             0         0    250.0  121.0   81.0  28.73       95.0   ␣
↪76.0
2             0         0    245.0  127.5   80.0  25.34       75.0   ␣
↪70.0
3             1         0    225.0  150.0   95.0  28.58       65.0  ␣
↪103.0
4             0         0    285.0  130.0   84.0  23.10       85.0   ␣
↪85.0

   TenYearCHD
0           0
1           0
2           0
3           1
4           0
```

[6]:
```
data.shape
```

[6]:
```
(4238, 16)
```

[7]:
```
data.describe()
```

[7]:
```
              male          age    education  currentSmoker  ␣
↪cigsPerDay  \
count  4238.000000  4238.000000  4133.000000    4238.000000  4209.000000
mean      0.429212    49.584946     1.978950       0.494101     9.003089
```

```
std        0.495022    8.572160    1.019791      0.500024    11.920094
min        0.000000   32.000000    1.000000      0.000000     0.000000
25%        0.000000   42.000000    1.000000      0.000000     0.000000
50%        0.000000   49.000000    2.000000      0.000000     0.000000
75%        1.000000   56.000000    3.000000      1.000000    20.000000
max        1.000000   70.000000    4.000000      1.000000    70.000000

            BPMeds  prevalentStroke  prevalentHyp     diabetes     ␣
 ↪totChol  \
count  4185.000000      4238.000000   4238.000000  4238.000000  4188.
 ↪000000
mean      0.029630         0.005899      0.310524     0.025720   236.
 ↪721585
std       0.169584         0.076587      0.462763     0.158316    44.
 ↪590334
min       0.000000         0.000000      0.000000     0.000000   107.
 ↪000000
25%       0.000000         0.000000      0.000000     0.000000   206.
 ↪000000
50%       0.000000         0.000000      0.000000     0.000000   234.
 ↪000000
75%       0.000000         0.000000      1.000000     0.000000   263.
 ↪000000
max       1.000000         1.000000      1.000000     1.000000   696.
 ↪000000

             sysBP        diaBP          BMI    heartRate      glucose  ␣
 ↪\
count  4238.000000  4238.000000  4219.000000  4237.000000  3850.000000
mean    132.352407    82.893464    25.802008    75.878924    81.966753
std      22.038097    11.910850     4.080111    12.026596    23.959998
min      83.500000    48.000000    15.540000    44.000000    40.000000
25%     117.000000    75.000000    23.070000    68.000000    71.000000
50%     128.000000    82.000000    25.400000    75.000000    78.000000
75%     144.000000    89.875000    28.040000    83.000000    87.000000
max     295.000000   142.500000    56.800000   143.000000   394.000000

         TenYearCHD
count  4238.000000
mean      0.151958
std       0.359023
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       1.000000
```

[8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4238 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             4238 non-null   int64  
 1   age              4238 non-null   int64  
 2   education        4133 non-null   float64
 3   currentSmoker    4238 non-null   int64  
 4   cigsPerDay       4209 non-null   float64
 5   BPMeds           4185 non-null   float64
 6   prevalentStroke  4238 non-null   int64  
 7   prevalentHyp     4238 non-null   int64  
 8   diabetes         4238 non-null   int64  
 9   totChol          4188 non-null   float64
 10  sysBP            4238 non-null   float64
 11  diaBP            4238 non-null   float64
 12  BMI              4219 non-null   float64
 13  heartRate        4237 non-null   float64
 14  glucose          3850 non-null   float64
 15  TenYearCHD       4238 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 529.9 KB
```

[9]:
```
# borrano una variable inecesaria
del data['education']
```

[10]:
```
#missing values
data.isnull().any()
```

[10]:
```
male               False
age                False
currentSmoker      False
cigsPerDay          True
BPMeds              True
prevalentStroke    False
prevalentHyp       False
diabetes           False
totChol             True
sysBP              False
diaBP              False
BMI                 True
heartRate           True
glucose             True
TenYearCHD         False
dtype: bool
```

[11]:
```
data.isnull().sum()
```

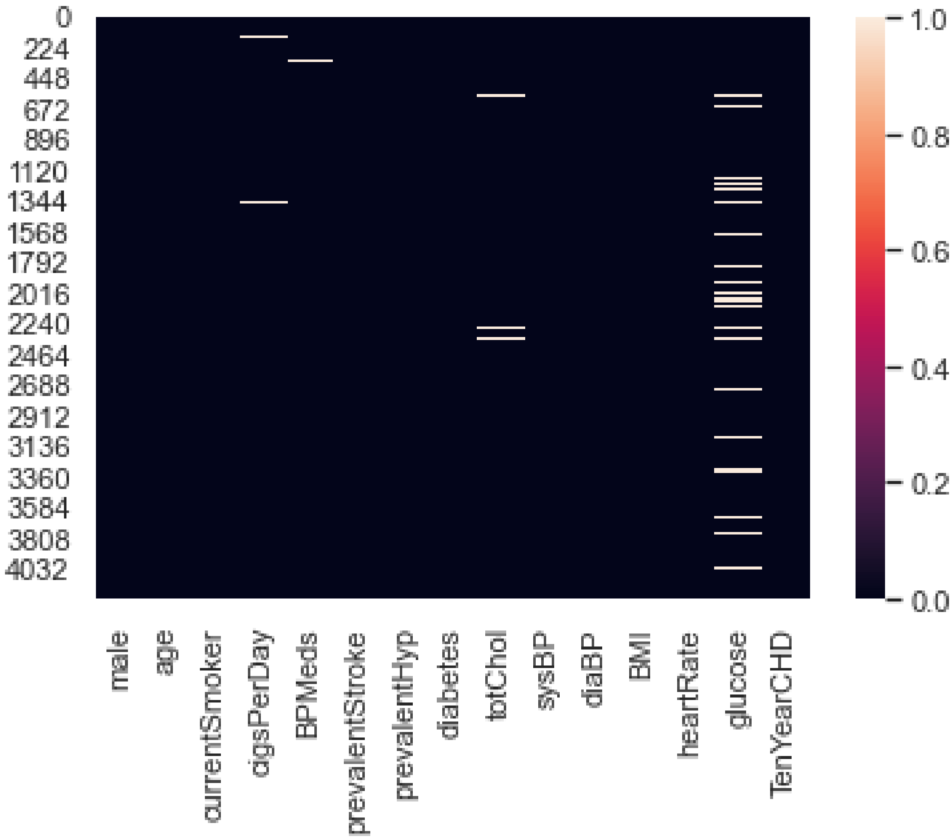
[11]:
```
male                 0
age                  0
currentSmoker        0
cigsPerDay          29
BPMeds              53
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             50
sysBP                0
diaBP                0
BMI                 19
heartRate            1
glucose            388
TenYearCHD           0
dtype: int64
```

[12]:
```
data.isnull().sum().sum()
```

[12]:
```
540
```

[13]:
```
sns.heatmap(data.isnull())
```

[13]:
```
<AxesSubplot:>
```



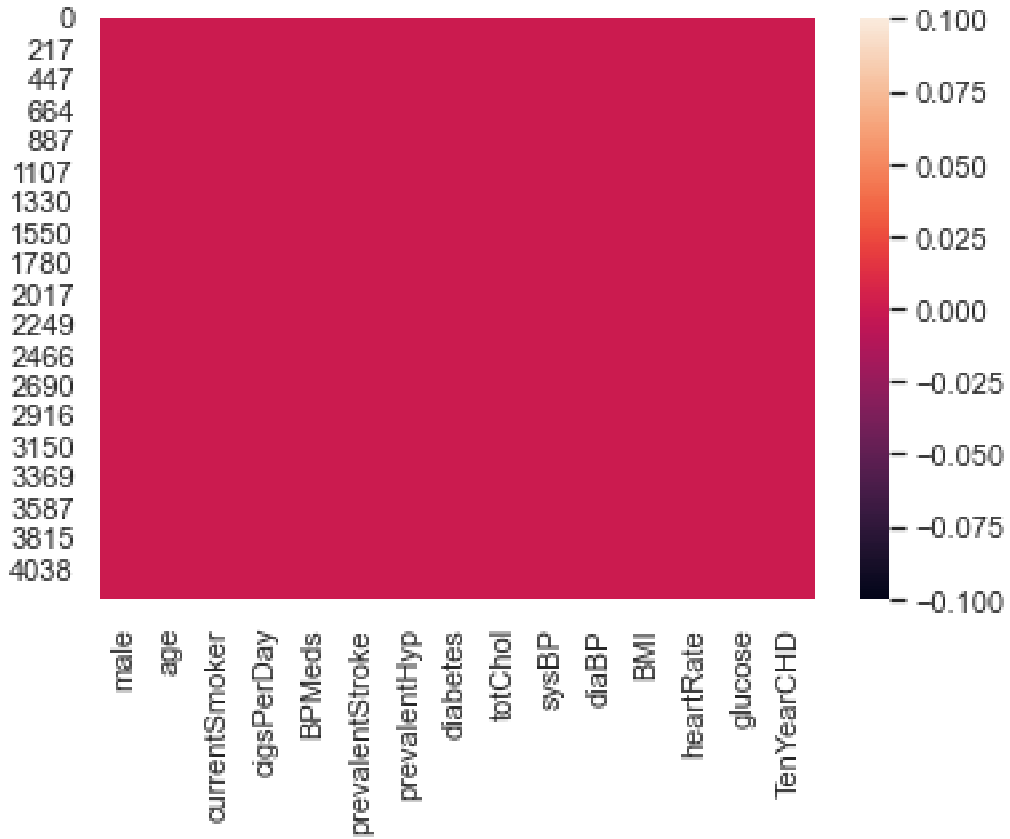
[14]:
```
data.dropna(axis=0,inplace=True)
```

[15]:
```
data.isnull().sum()
```

[15]:
```
male               0
age                0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
TenYearCHD         0
dtype: int64
```

[16]:
```
sns.heatmap(data.isnull())
```

[16]: <AxesSubplot:>



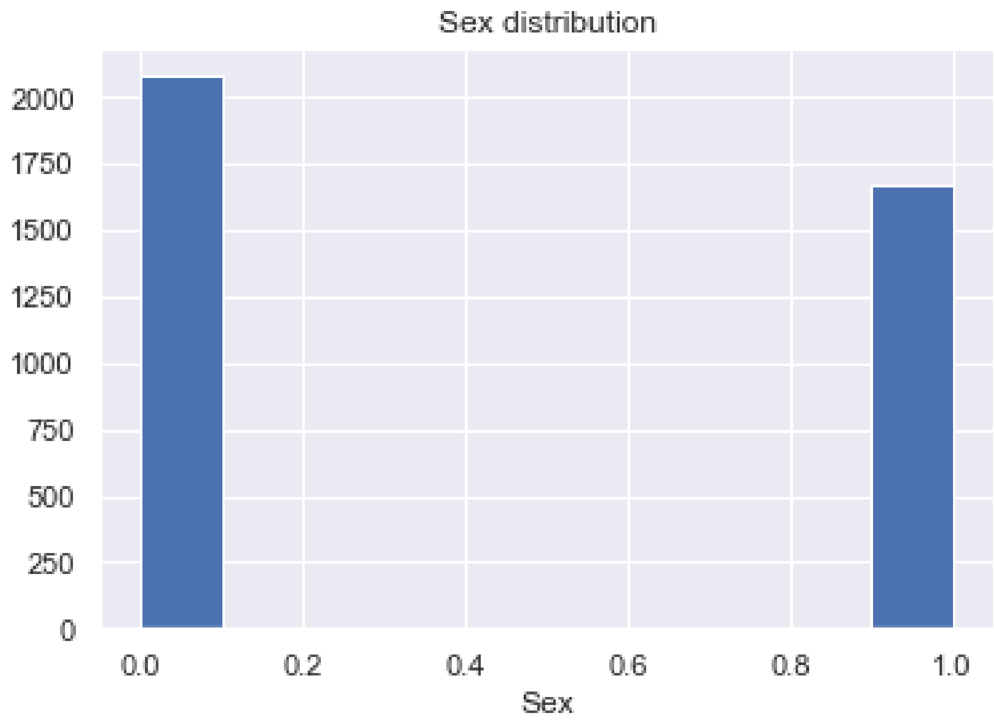
[17]:
```python
# duplicated values
data.duplicated().any()
```

[17]: False

Exploratory Analysis

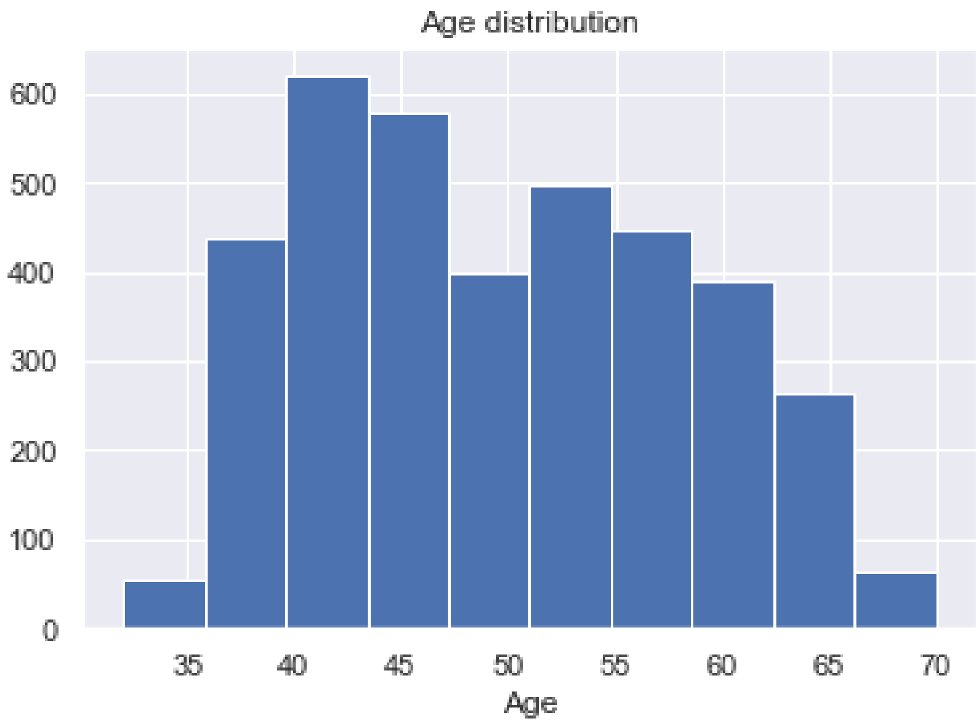
[18]:
```python
# Sex distibution
plt.hist(data['male'])
plt.title('Sex distribution')
plt.xlabel('Sex')
```

[18]: Text(0.5, 0, 'Sex')



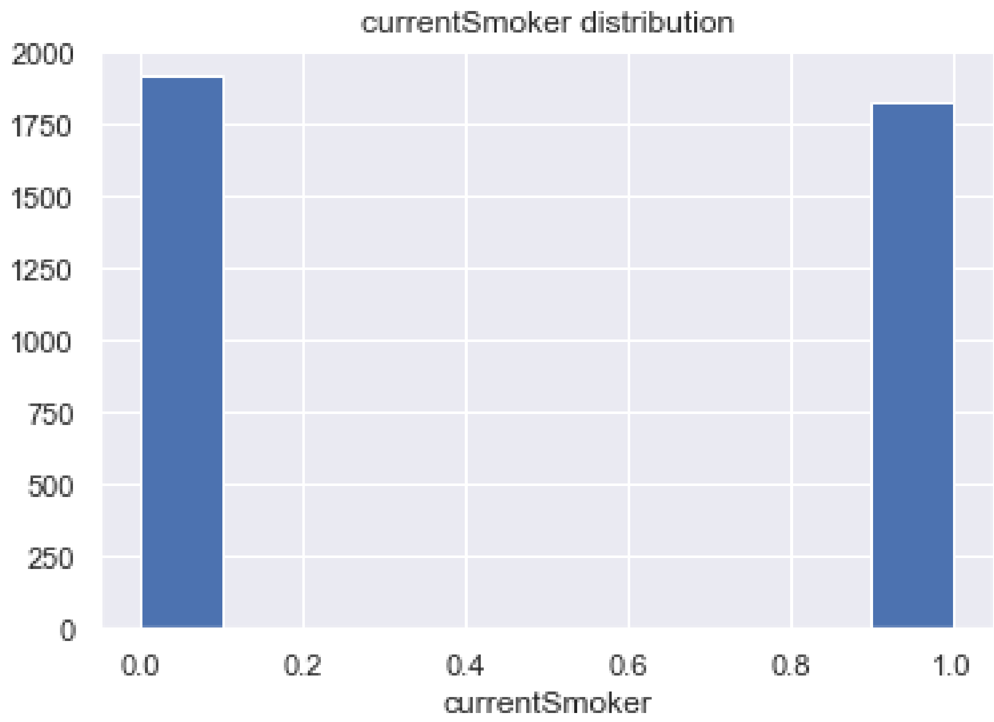
[19]:
```
# Age distribution
plt.hist(data['age'])
plt.title('Age distribution')
plt.xlabel('Age')
```

[19]:
```
Text(0.5, 0, 'Age')
```



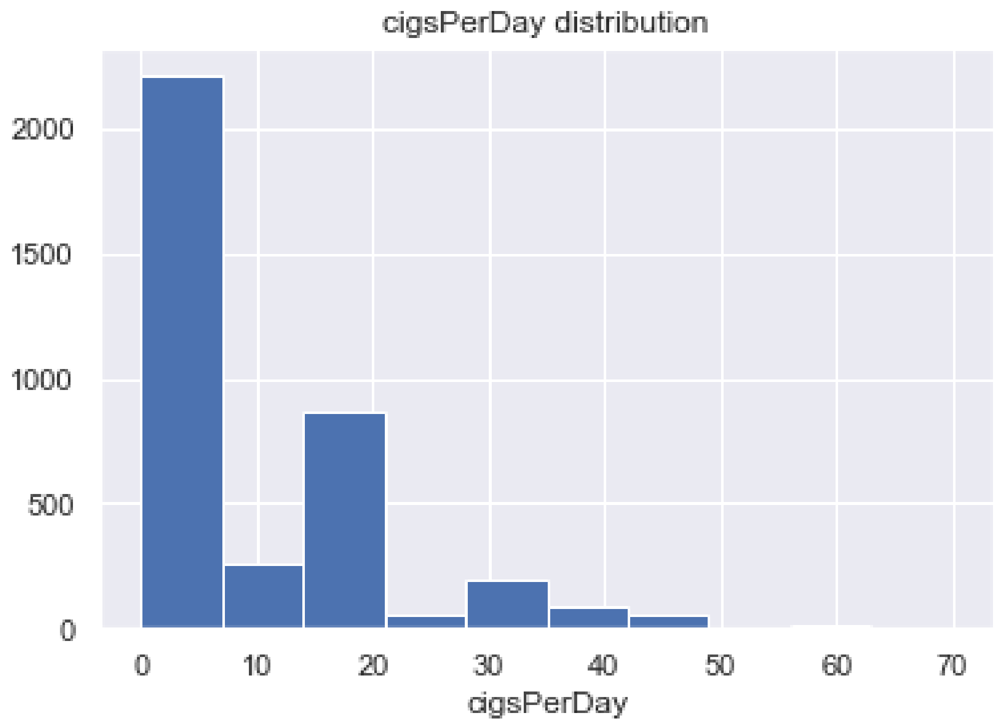
[20]:
```
# currentSmoker distribution
plt.hist(data['currentSmoker'])
plt.title('currentSmoker distribution')
plt.xlabel('currentSmoker')
```

[20]:
```
Text(0.5, 0, 'currentSmoker')
```



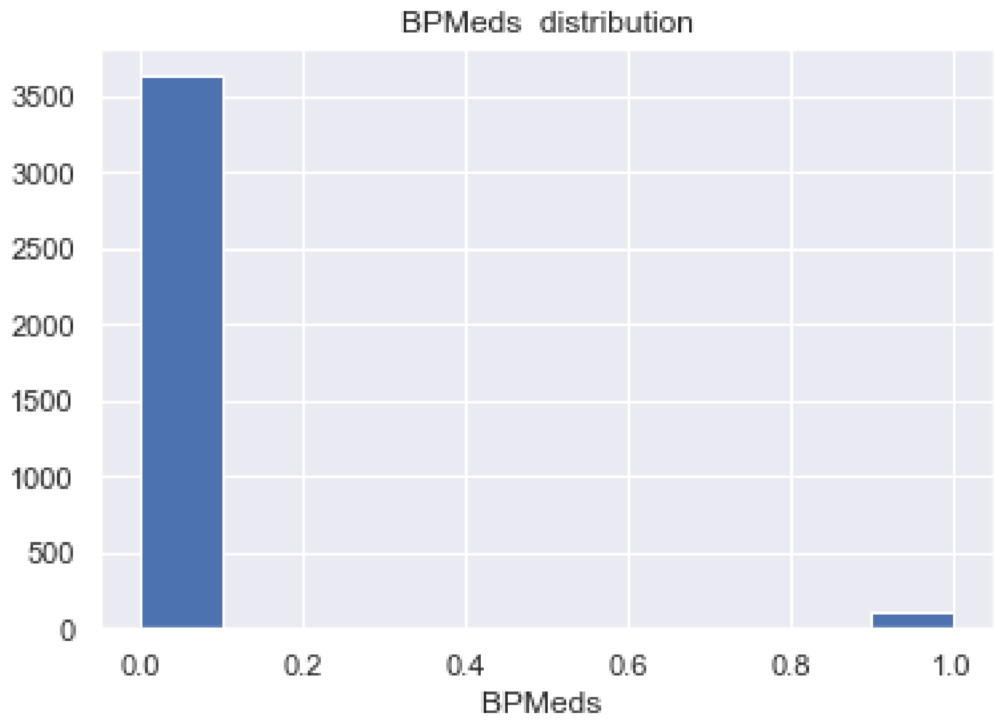
[21]:
```
# cigsPerDay distribution
plt.hist(data['cigsPerDay'])
plt.title('cigsPerDay distribution')
plt.xlabel('cigsPerDay')
```

[21]: Text(0.5, 0, 'cigsPerDay')



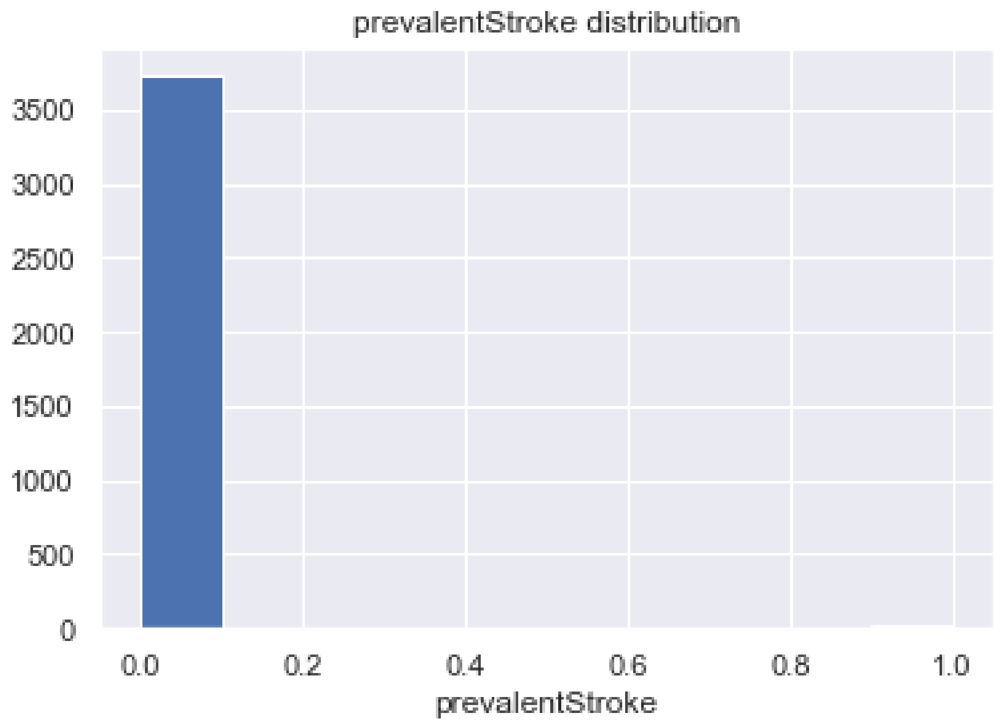
[22]:
```
# BPMeds  distribution
plt.hist(data['BPMeds'])
plt.title('BPMeds  distribution')
plt.xlabel('BPMeds ')
```

[22]: Text(0.5, 0, 'BPMeds ')



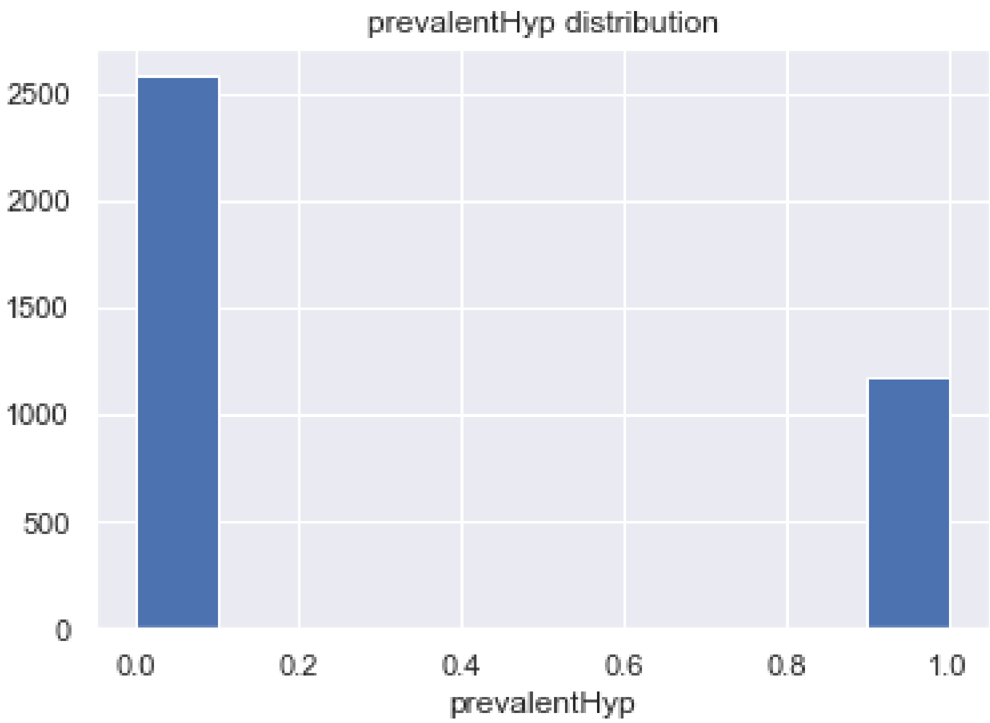
[23]:
```
# prevalentStroke distribution
plt.hist(data['prevalentStroke'])
plt.title('prevalentStroke distribution')
plt.xlabel('prevalentStroke')
```

[23]: Text(0.5, 0, 'prevalentStroke')



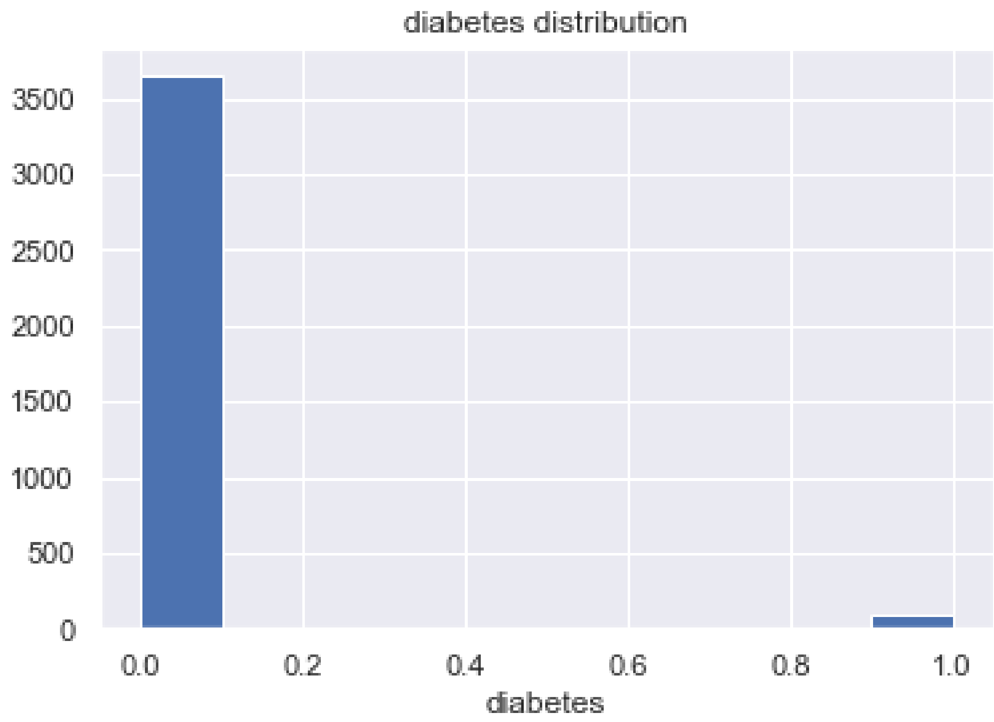
[24]:
```
# prevalentHyp distribution
plt.hist(data['prevalentHyp'])
plt.title('prevalentHyp distribution')
plt.xlabel('prevalentHyp')
```

[24]: Text(0.5, 0, 'prevalentHyp')



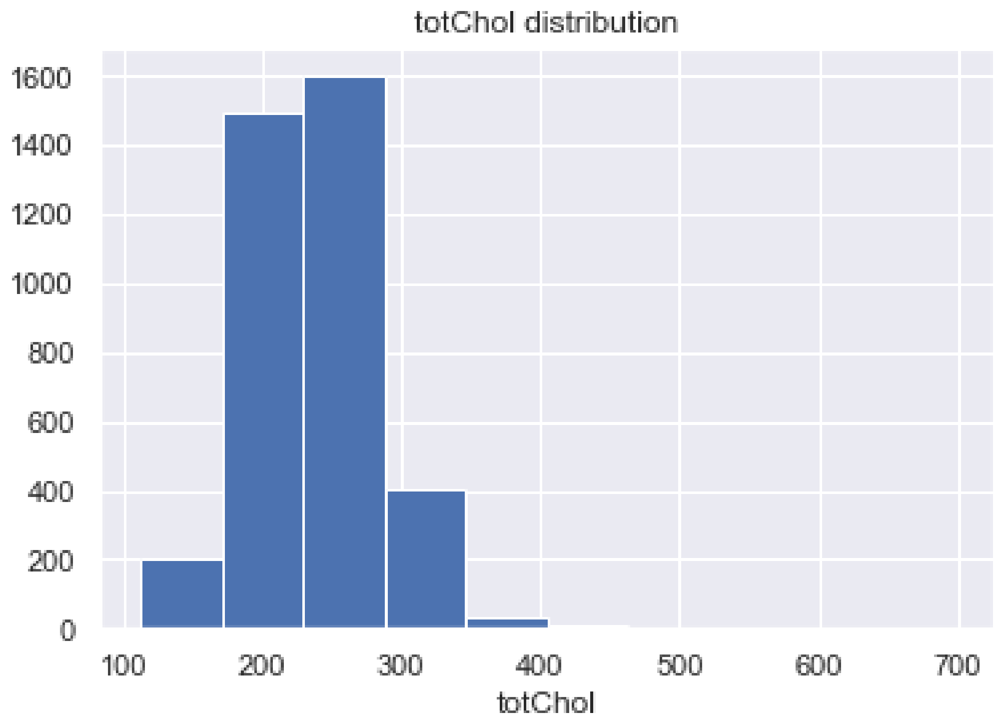
[25]:
```
# diabetes distribution
plt.hist(data['diabetes'])
plt.title('diabetes distribution')
plt.xlabel('diabetes')
```

[25]: `Text(0.5, 0, 'diabetes')`



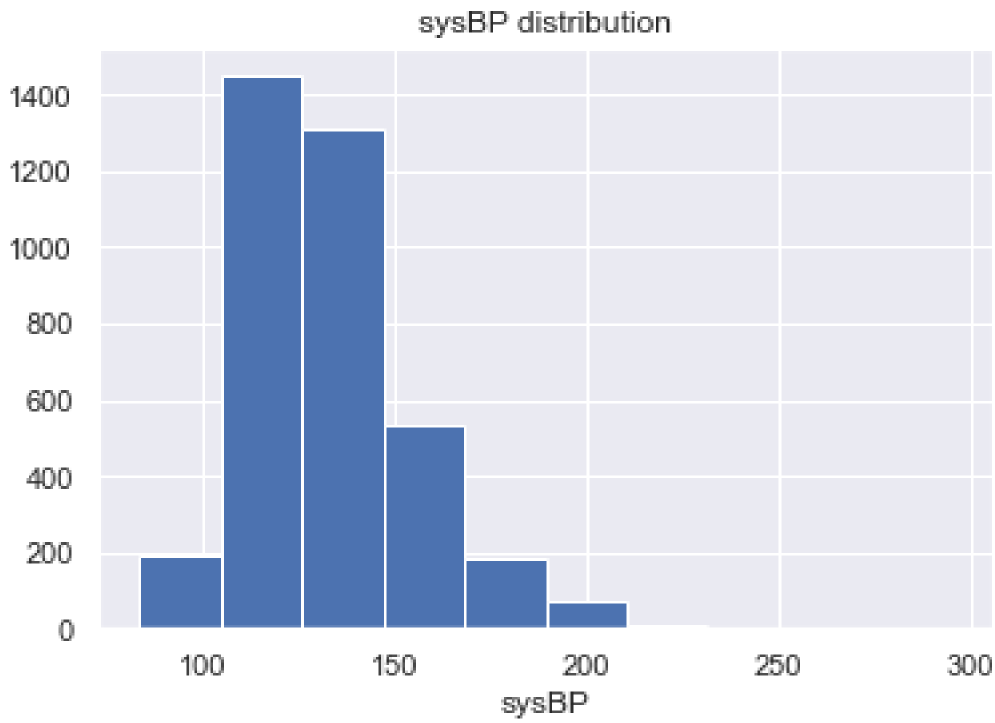
[26]:
```
# totChol distribution
plt.hist(data['totChol'])
plt.title('totChol distribution')
plt.xlabel('totChol')
```

[26]:
```
Text(0.5, 0, 'totChol')
```



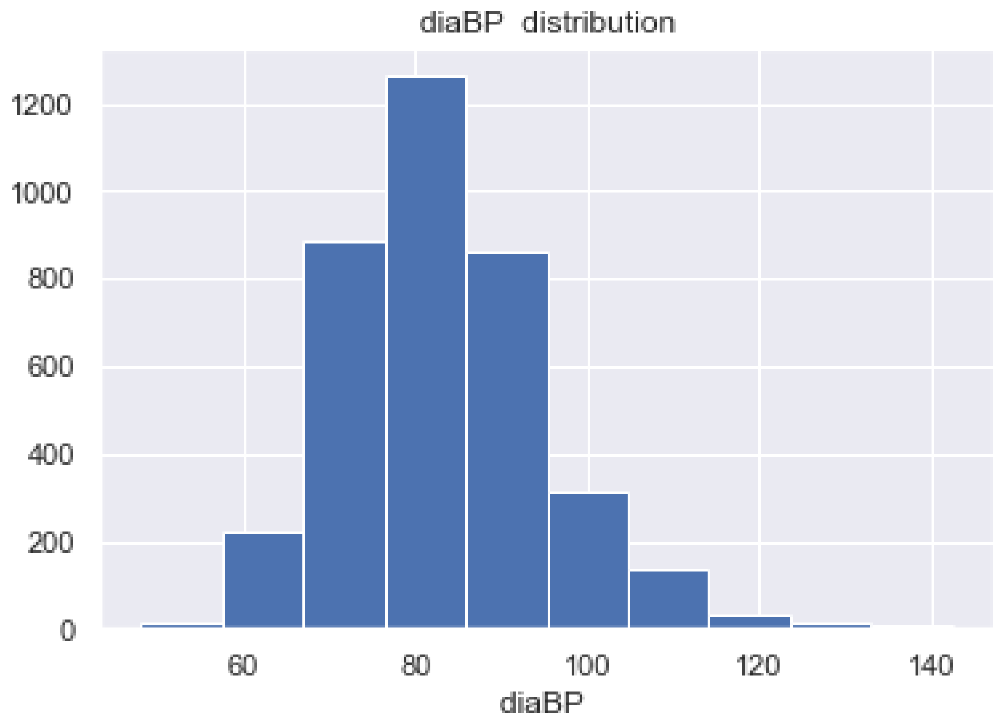
[27]:
```
# sysBP distribution
plt.hist(data['sysBP'])
plt.title('sysBP distribution')
plt.xlabel('sysBP')
```

[27]:
```
Text(0.5, 0, 'sysBP')
```



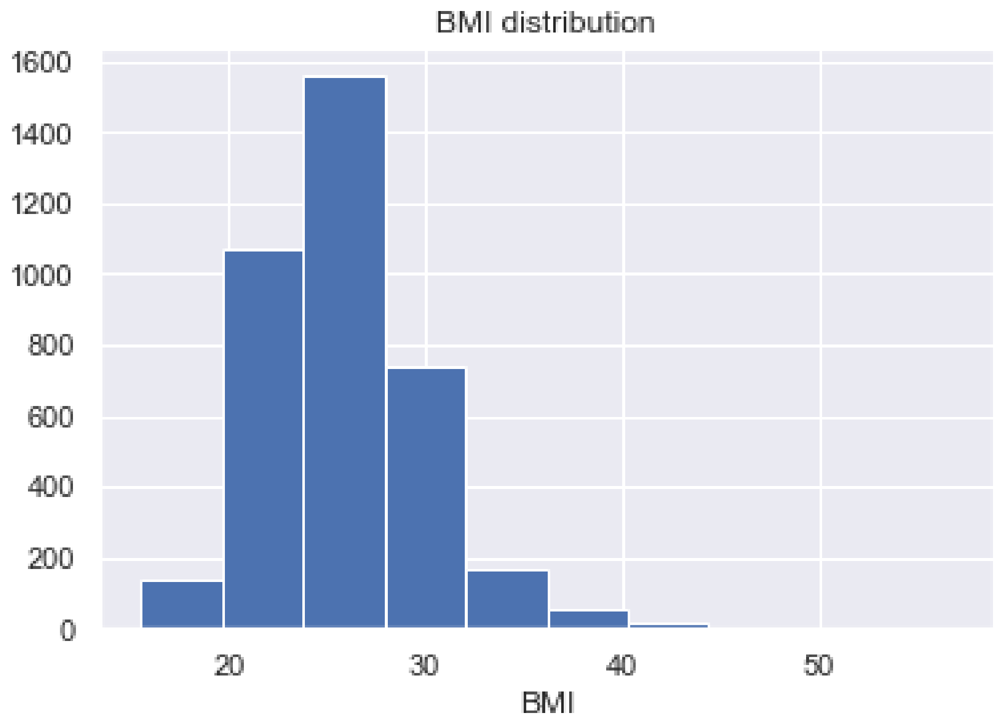
[28]:
```
# diaBP  distribution
plt.hist(data['diaBP'])
plt.title('diaBP  distribution')
plt.xlabel('diaBP ')
```

[28]: Text(0.5, 0, 'diaBP ')



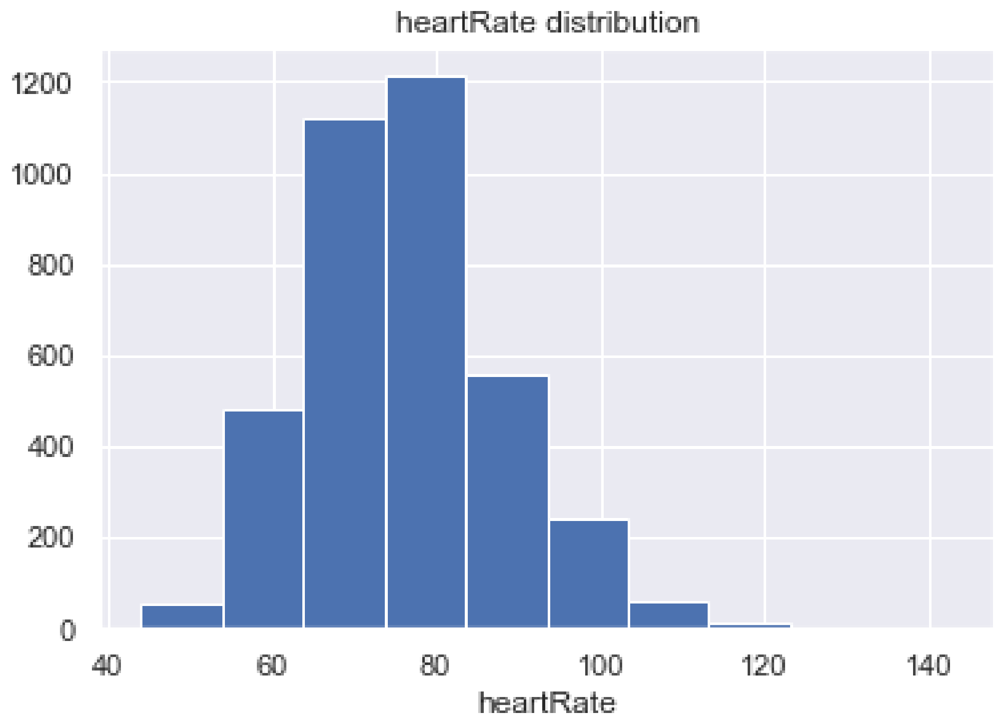
[29]:
```
# BMI distribution
plt.hist(data['BMI'])
plt.title('BMI distribution')
plt.xlabel('BMI')
```

[29]:
```
Text(0.5, 0, 'BMI')
```



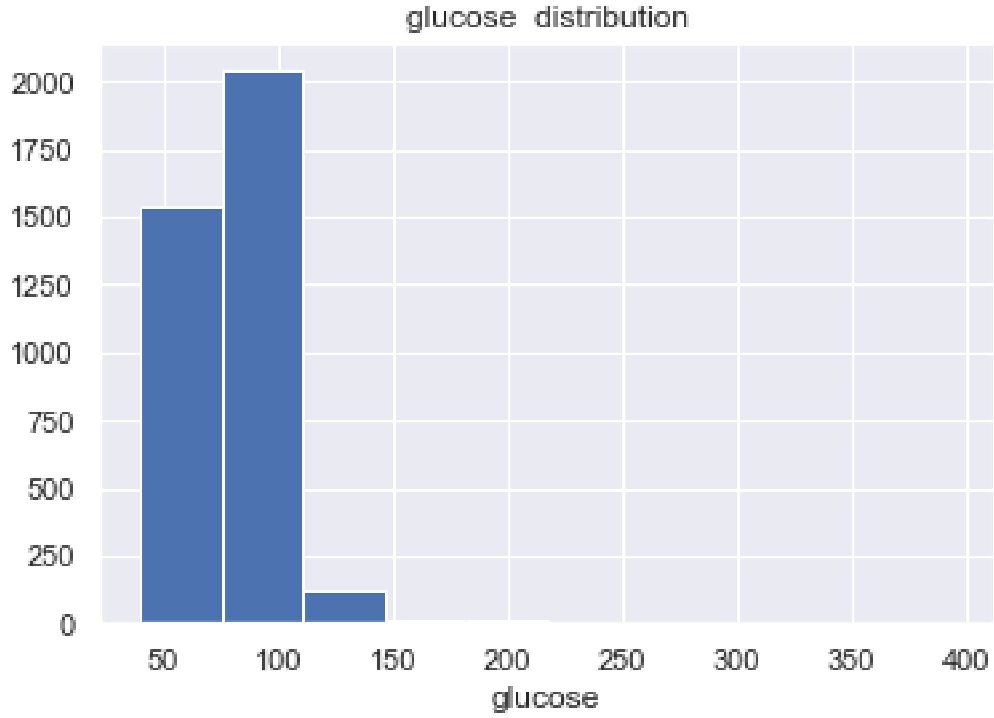
[30]:
```
# heartRate distribution
plt.hist(data['heartRate'])
plt.title('heartRate distribution')
plt.xlabel('heartRate')
```

[30]: Text(0.5, 0, 'heartRate')



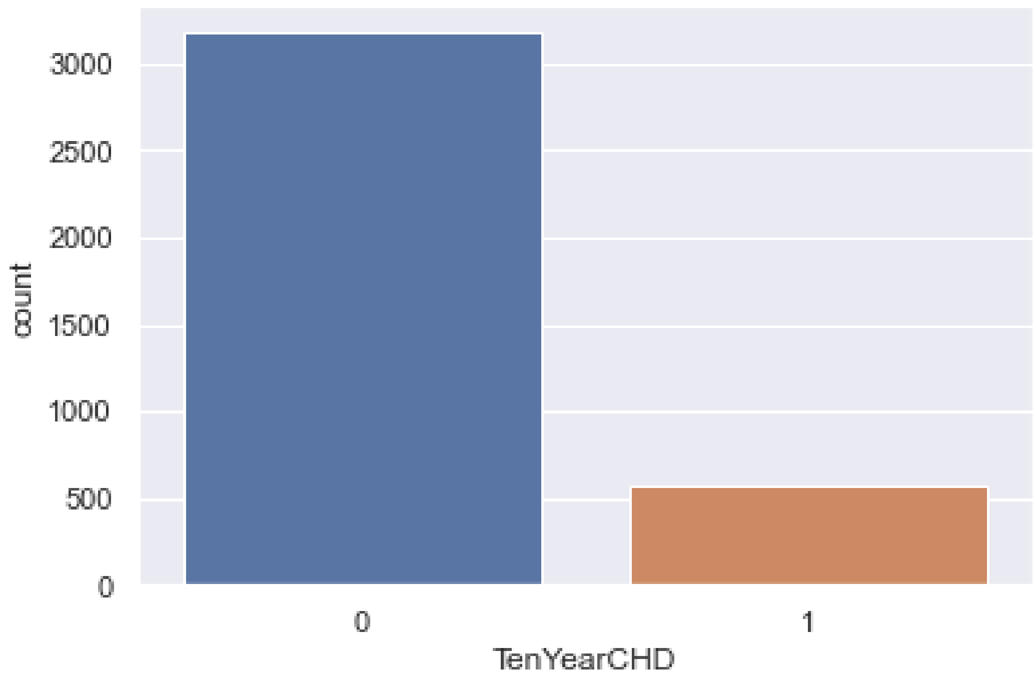
[31]:
```
# glucose  distribution
plt.hist(data['glucose'])
plt.title('glucose  distribution')
plt.xlabel('glucose')
```

[31]: `Text(0.5, 0, 'glucose')`



[32]:
```
# TenYearCHD distribution
sns.countplot(x='TenYearCHD',data=data)
```

[32]: <AxesSubplot:xlabel='TenYearCHD', ylabel='count'>



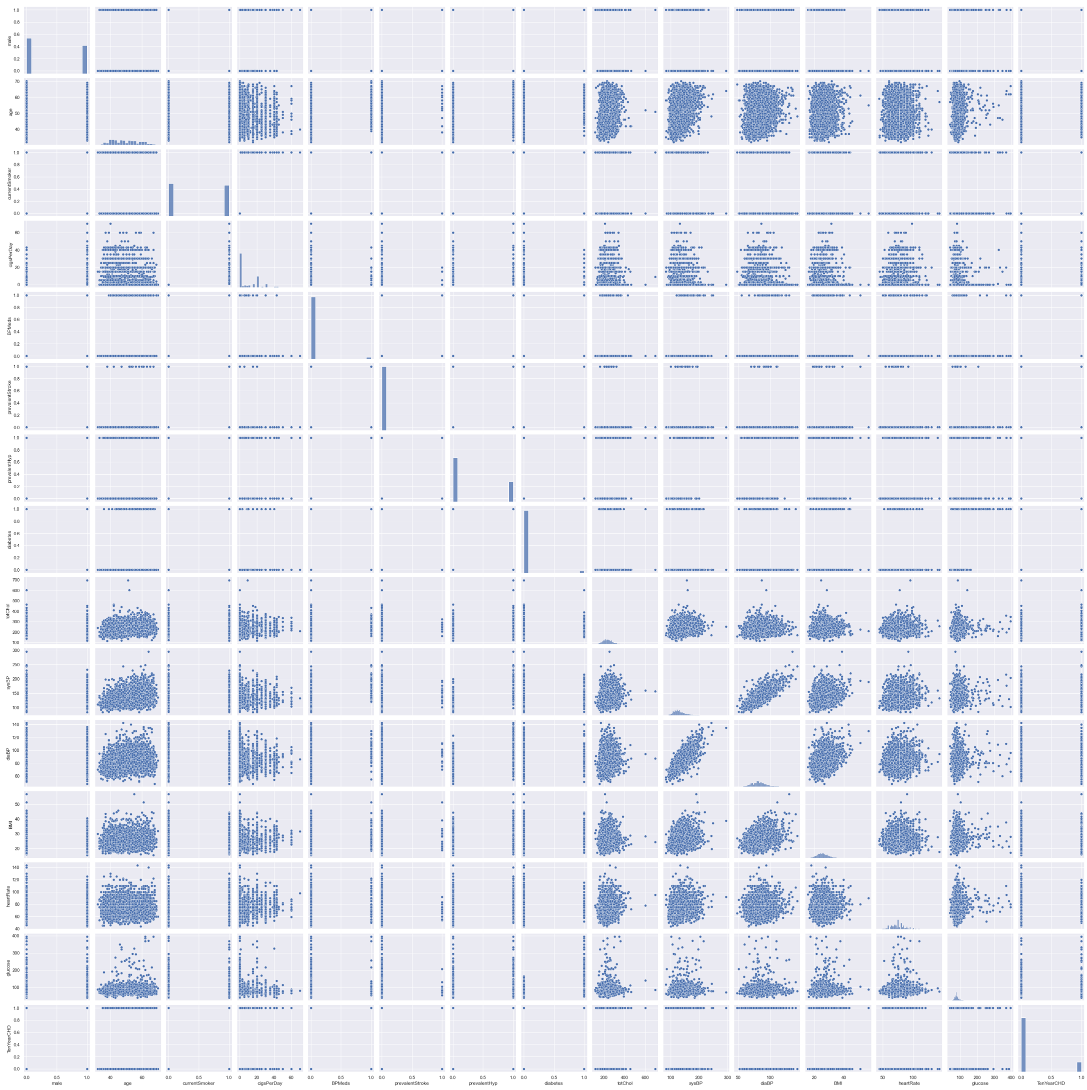
[33]:
```
data.TenYearCHD.value_counts()
```

[33]:
```
0    3177
1     572
Name: TenYearCHD, dtype: int64
```

There are 3,177 patents without heart disease and 572 patients at risk of heart disease.

[35]:
```
# all variables vs all variables plot
sns.pairplot(data=data)
```

[35]:
```
<seaborn.axisgrid.PairGrid at 0x24b4f721bb0>
```



[37]:
```
correlacion = data.corr()
```

[38]:
```
correlacionlacion
```

[38]:

```
                    male       age  currentSmoker  cigsPerDay  ␣
↪BPMeds  \
male            1.000000 -0.024120       0.203861    0.326780 -0.052359
age            -0.024120  1.000000      -0.211427   -0.188611  0.131629
currentSmoker   0.203861 -0.211427       1.000000    0.773166 -0.051828
cigsPerDay      0.326780 -0.188611       0.773166    1.000000 -0.046601
BPMeds         -0.052359  0.131629      -0.051828   -0.046601  1.000000
prevalentStroke -0.002509  0.049990      -0.037582   -0.035711  0.111595
prevalentHyp    0.002987  0.305735      -0.104753   -0.066911  0.263089
diabetes        0.011847  0.109257      -0.045319   -0.039411  0.056322
totChol        -0.067506  0.260967      -0.050025   -0.030427  0.089554
sysBP          -0.044638  0.388558      -0.133098   -0.092292  0.269507
diaBP           0.053602  0.205774      -0.113915   -0.056108  0.199400
BMI             0.074630  0.136093      -0.165165   -0.090032  0.105090
heartRate      -0.115091 -0.005857       0.054545    0.066726  0.010232
glucose         0.003236  0.118426      -0.054180   -0.055165  0.052442
TenYearCHD      0.096056  0.231414       0.021722    0.056064  0.084704

                 prevalentStroke  prevalentHyp  diabetes   totChol     ␣
↪sysBP  \
male                   -0.002509      0.002987  0.011847 -0.067506 -0.
↪044638
age                     0.049990      0.305735  0.109257  0.260967  0.
↪388558
currentSmoker          -0.037582     -0.104753 -0.045319 -0.050025 -0.
↪133098
cigsPerDay             -0.035711     -0.066911 -0.039411 -0.030427 -0.
↪092292
BPMeds                  0.111595      0.263089  0.056322  0.089554  0.
↪269507
prevalentStroke         1.000000      0.065208  0.009417  0.012259  0.
↪060431
prevalentHyp            0.065208      1.000000  0.082096  0.165049  0.
↪697960
diabetes                0.009417      0.082096  1.000000  0.047374  0.
↪104415
totChol                 0.012259      0.165049  0.047374  1.000000  0.
↪216572
sysBP                   0.060431      0.697960  0.104415  0.216572  1.
↪000000
diaBP                   0.055232      0.616655  0.051841  0.170353  0.
↪785909
BMI                     0.035550      0.303382  0.093061  0.119398  0.
↪330569
heartRate              -0.016675      0.142512  0.063383  0.094802  0.
↪181482
```

```
glucose                    0.015779       0.085959  0.616084  0.046769  0.
 ↪132928
TenYearCHD                 0.047669       0.178779  0.093190  0.089408  0.
 ↪220170

                    diaBP       BMI  heartRate   glucose  TenYearCHD
male             0.053602  0.074630  -0.115091  0.003236    0.096056
age              0.205774  0.136093  -0.005857  0.118426    0.231414
currentSmoker   -0.113915 -0.165165   0.054545 -0.054180    0.021722
cigsPerDay      -0.056108 -0.090032   0.066726 -0.055165    0.056064
BPMeds           0.199400  0.105090   0.010232  0.052442    0.084704
prevalentStroke  0.055232  0.035550  -0.016675  0.015779    0.047669
prevalentHyp     0.616655  0.303382   0.142512  0.085959    0.178779
diabetes         0.051841  0.093061   0.063383  0.616084    0.093190
totChol          0.170353  0.119398   0.094802  0.046769    0.089408
sysBP            0.785909  0.330569   0.181482  0.132928    0.220170
diaBP            1.000000  0.384166   0.175175  0.061891    0.149206
BMI              0.384166  1.000000   0.071953  0.088121    0.084278
heartRate        0.175175  0.071953   1.000000  0.099528    0.022668
glucose          0.061891  0.088121   0.099528  1.000000    0.124071
TenYearCHD       0.149206  0.084278   0.022668  0.124071    1.000000
```

We look for variables that have a greater correlation with our dependent variable so that, initially, we can do a simple logistic regression with an independent variable and our dependent variable. In the case of the simple logistic regression model, we are going to use age as an independent variable and we will see what results it gives us, since it is a variable that has a fairly high correlation.

Logistic Regression

[39]:
```
# Simple logistic regression
# independent variable
X = data[['age']]
```

[40]:
```
# dependent varieable
y = data['TenYearCHD']
```

[41]:
```
X.head()
```

[41]:

	age
0	39
1	46
2	48
3	61
4	46

[42]:
```
y.head()
```

[42]:
```
0    0
1    0
2    0
3    1
4    0
Name: TenYearCHD, dtype: int64
```

[43]:
```
from sklearn.linear_model import LogisticRegression
```

[44]:
```
logit_reg = LogisticRegression()
logit_reg.fit(X,y)
```

[44]:
```
LogisticRegression()
```

[45]:
```
# beta_1
logit_reg.coef_
```

[45]:
```
array([[0.07670223]])
```

[46]:
```
# beta 0
logit_reg.intercept_
```

[46]:
```
array([-5.66458369])
```

Second method

[47]:
```
X_cons = sm.add_constant(X)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\tsatools.py:
↪142:
FutureWarning: In a future version of pandas all arguments of concat␣
↪except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

[48]:
```
X_cons.head()
```

[48]:

	const	age
0	1.0	39
1	1.0	46
2	1.0	48
3	1.0	61
4	1.0	46

[49]:
```
import statsmodels.discrete.discrete_model as smd
```

[50]:
```
logit = smd.Logit(y, X_cons).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.400327
         Iterations 6
```

[51]:
```
logit.summary()
```

[51]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:             TenYearCHD   No. Observations:                    ␣
 ↪  3749
Model:                          Logit   Df Residuals:                        ␣
 ↪  3747
Method:                           MLE   Df Model:                            ␣
 ↪     1
Date:                Wed, 29 Sep 2021   Pseudo R-squ.:                      ␣
 ↪0.06279
Time:                        17:41:12   Log-Likelihood:                     ␣
 ↪-1500.8
converged:                       True   LL-Null:                            ␣
 ↪-1601.4
Covariance Type:            nonrobust   LLR p-value:                      1.
 ↪198e-45
==============================================================================
                 coef    std err          z      P>|z|      [0.025           ␣
 ↪0.975]
------------------------------------------------------------------------------
const         -5.6647      0.303    -18.698      0.000      -6.258           ␣
 ↪-5.071
age            0.0767      0.006     13.663      0.000       0.066           ␣
 ↪ 0.088
==============================================================================
"""
```

Since the P value is very low, this shows us a significantly high statistical relationship with the probability of having heart disease, but it is an insufficient analysis up to this point, so we will continue the analysis to improve our model, doing a multiple logistic regression and a Linear Discriminant Analysis

Multiple logistic regression

[52]:
```
X = data.loc[:,data.columns != 'TenYearCHD']
```

[53]:
```
y = data['TenYearCHD']
```

[54]:
```
mul_lr = LogisticRegression()
mul_lr.fit(X,y)
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning:␣
↪lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown␣
↪in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[54]:
```
LogisticRegression()
```

[55]:
```
# beta_1 values
mul_lr.coef_
```

[55]:
```
array([[ 0.53119444,  0.02931899, -0.24920643,  0.01984001,  0.15962647,
         0.05725021,  1.01702757,  0.17800746, -0.00127438,  0.0137023 ,
        -0.03091462, -0.04551082, -0.02208319,  0.00471502]])
```

[56]:
```
# beta_0 values
mul_lr.intercept_
```

[56]:
```
array([-0.42036314])
```

[57]:
```
X_cons = sm.add_constant(X)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\tsatools.py:
↪142:
FutureWarning: In a future version of pandas all arguments of concat␣
↪except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

[58]:
```
X_cons.head()
```

[58]:
```
   const  male  age  currentSmoker  cigsPerDay  BPMeds  prevalentStroke␣
↪ \
0    1.0     1   39              0         0.0     0.0                0
1    1.0     0   46              0         0.0     0.0                0
```

```
2    1.0     1   48              1        20.0     0.0              0
3    1.0     0   61              1        30.0     0.0              0
4    1.0     0   46              1        23.0     0.0              0

   prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  heartRate  ␣
 ↪glucose
0             0         0    195.0  106.0   70.0  26.97       80.0   ␣
 ↪77.0
1             0         0    250.0  121.0   81.0  28.73       95.0   ␣
 ↪76.0
2             0         0    245.0  127.5   80.0  25.34       75.0   ␣
 ↪70.0
3             1         0    225.0  150.0   95.0  28.58       65.0  ␣
 ↪103.0
4             0         0    285.0  130.0   84.0  23.10       85.0   ␣
 ↪85.0
```

[59]:
```
logit = smd.Logit(y, X_cons).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.377199
         Iterations 7
```

[60]:
```
logit.summary()
```

[60]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              TenYearCHD   No. Observations:                   ␣
 ↪  3749
Model:                           Logit   Df Residuals:                       ␣
 ↪  3734
Method:                            MLE   Df Model:                           ␣
 ↪    14
Date:                 Wed, 29 Sep 2021   Pseudo R-squ.:                      ␣
 ↪0.1169
Time:                         17:42:44   Log-Likelihood:                    ␣
 ↪-1414.1
converged:                        True   LL-Null:                           ␣
 ↪-1601.4
Covariance Type:             nonrobust   LLR p-value:                     2.
 ↪922e-71
================================================================================
===
                      coef    std err          z      P>|z|      [0.025
0.975]
```

```
------------------------------------------------------------------------------
---
const             -8.6463      0.687    -12.577      0.000      -9.994
-7.299
male               0.5740      0.107      5.343      0.000       0.363
0.785
age                0.0640      0.007      9.787      0.000       0.051
0.077
currentSmoker      0.0732      0.155      0.473      0.636      -0.230
0.376
cigsPerDay         0.0184      0.006      3.003      0.003       0.006
0.030
BPMeds             0.1446      0.232      0.622      0.534      -0.311
0.600
prevalentStroke    0.7191      0.489      1.471      0.141      -0.239
1.677
prevalentHyp       0.2146      0.136      1.574      0.116      -0.053
0.482
diabetes           0.0025      0.312      0.008      0.994      -0.609
0.614
totChol            0.0022      0.001      2.074      0.038       0.000
0.004
sysBP              0.0153      0.004      4.080      0.000       0.008
0.023
diaBP             -0.0039      0.006     -0.619      0.536      -0.016
0.009
BMI                0.0103      0.013      0.820      0.412      -0.014
0.035
heartRate         -0.0023      0.004     -0.550      0.583      -0.010
0.006
glucose            0.0076      0.002      3.408      0.001       0.003
0.012
==============================================================================
===
"""
```

[124]:
```
params = np.exp(logit.params)
conf = np.exp(logit.conf_int())
conf['OR'] = params
pvalue=round(logit.pvalues,3)
conf['pvalue']=pvalue
conf.columns = ['CI 95%(2.5%)', 'CI 95%(97.5%)', 'Odds Ratio','pvalue']
print ((conf))
```

```
                 CI 95%(2.5%)  CI 95%(97.5%)  Odds Ratio  pvalue
const                0.000046       0.000676    0.000176   0.000
male                 1.438236       2.191467    1.775344   0.000
age                  1.052554       1.079902    1.066140   0.000
```

currentSmoker	0.794680	1.456670	1.075912	0.636
cigsPerDay	1.006399	1.030829	1.018541	0.003
BPMeds	0.732958	1.821749	1.155537	0.534
prevalentStroke	0.787199	5.351717	2.052527	0.141
prevalentHyp	0.948707	1.619173	1.239403	0.116
diabetes	0.543768	1.848131	1.002474	0.994
totChol	1.000124	1.004373	1.002246	0.038
sysBP	1.008006	1.022975	1.015463	0.000
diaBP	0.983737	1.008566	0.996074	0.536
BMI	0.985837	1.035407	1.010318	0.412
heartRate	0.989621	1.005880	0.997718	0.583
glucose	1.003225	1.012004	1.007605	0.001

The probability of being diagnosed with heart disease in the case of male is higher by 77.5

For age, we can see, keeping all the other variables constant that, for each year that a person is older, there is a 6.6

We can see that, for the case in which the number of cigarettes per day increases, we have an increase of 1.85on the CDH odds.

For systolic blood pressure we can see that we have a 1.5

Prediction y confusion matrix

[61]:
```
mul_lr.predict_proba(X)
```

[61]:
```
array([[0.91188244, 0.08811756],
       [0.96385772, 0.03614228],
       [0.86436731, 0.13563269],
       ...,
       [0.65921324, 0.34078676],
       [0.69113227, 0.30886773],
       [0.9015448 , 0.0984552 ]])
```

[62]:
```
y_pred = mul_lr.predict(X)
```

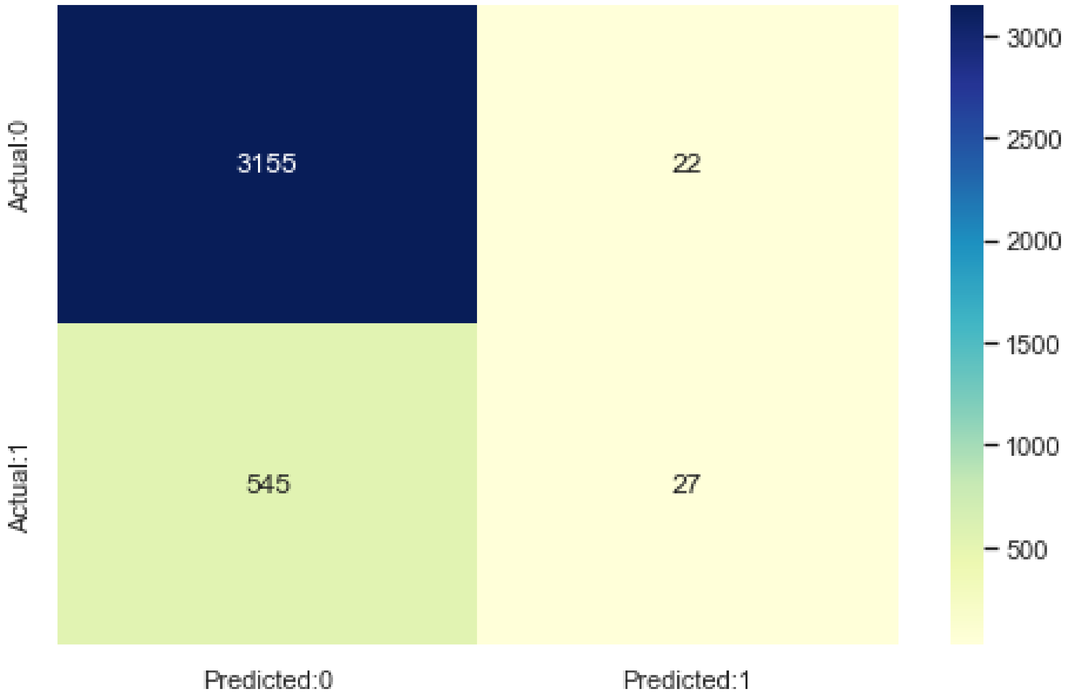
[63]:
```
y_pred
```

[63]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

[65]:
```
# make the fconfusion matrix
cm = confusion_matrix(y,y_pred)
```

[67]:
```
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:
↪1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu")
```

[67]: <AxesSubplot:>



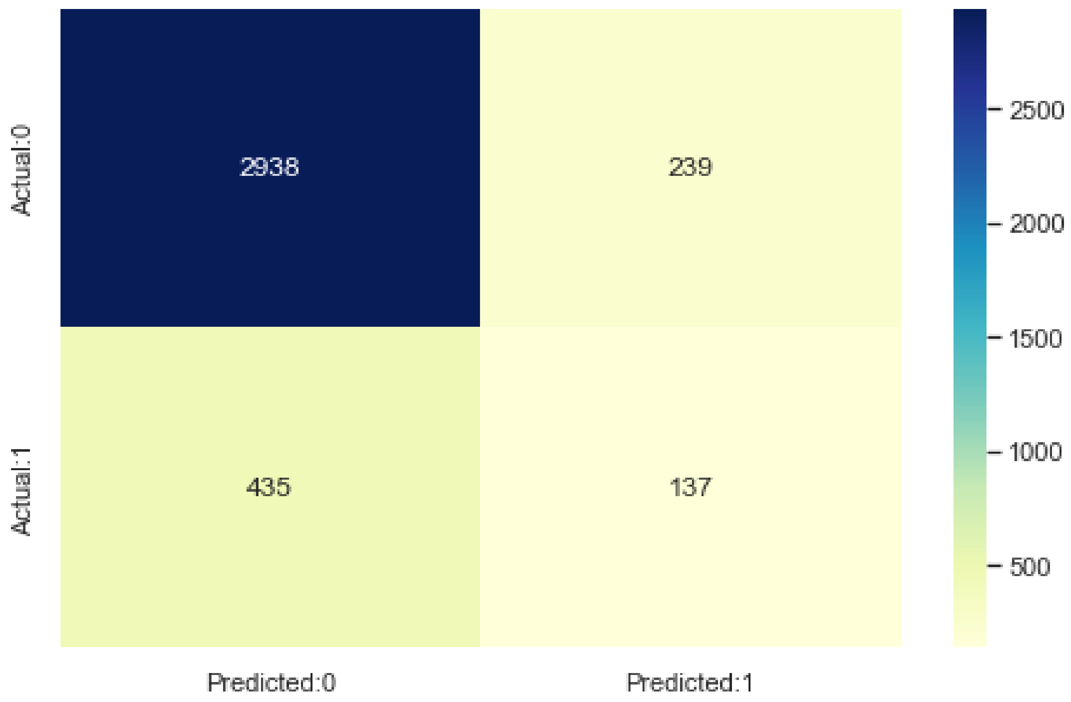
[64]:
```
#make a condition for 0.3, for values greater than 30%
y_pred_03 = (mul_lr.predict_proba(X)[:,1] >= 0.3)

y_pred_03
```

[64]: array([False, False, False, ..., True, True, False])

[69]:
```
# confusion matrix of the last condition
cm30 = confusion_matrix(y,y_pred_03)
```

[70]:
```
conf_matrix=pd.DataFrame(data=cm30,columns=['Predicted:0','Predicted:
 ↪1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu")
```

[70]: <AxesSubplot:>



[72]:
```
# evaluation of model performance and prediction probability
from sklearn.metrics import precision_score, recall_score
```

[73]:
```
precision_score(y, y_pred)
```

[73]: 0.5510204081632653

[74]:
```
recall_score(y,y_pred)
```

[74]: 0.0472027972027972

[75]:
```
from sklearn.metrics import roc_auc_score
```

[76]:
```
roc_auc_score(y, y_pred)
```

[76]: 0.5201390127027522

Linear Discriminant Analysis

[77]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

[78]:
```
#model
mul_lda = LinearDiscriminantAnalysis()
mul_lda.fit(X,y)
```

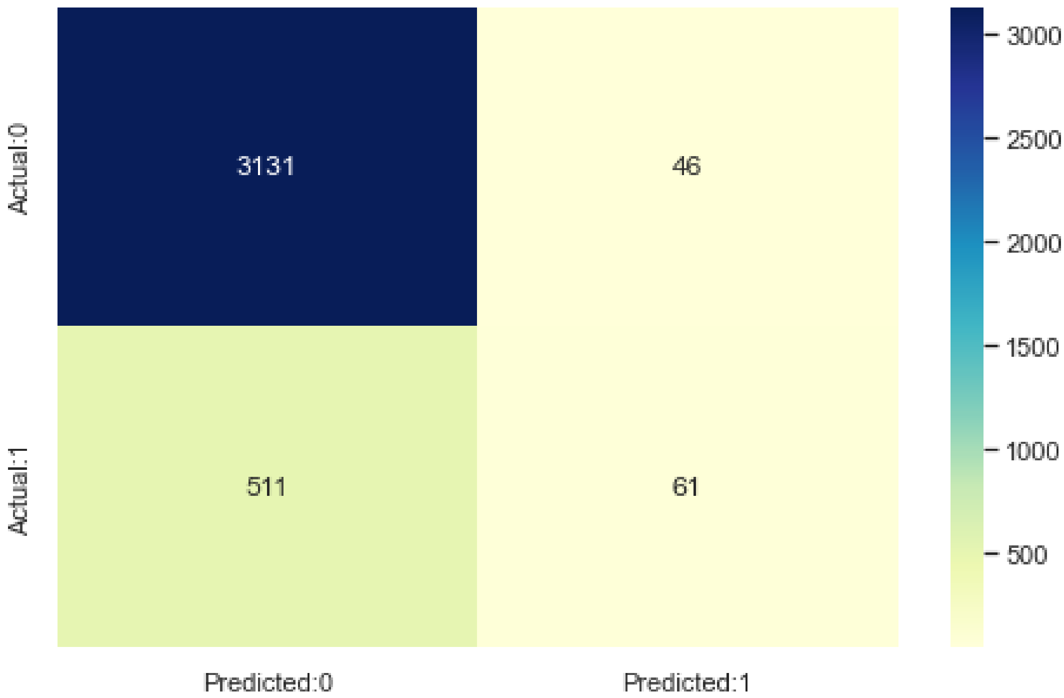
[78]: LinearDiscriminantAnalysis()

[79]:
```
# model prediction
y_pred_lda = mul_lda.predict(X)
```

[80]:
```
# confusion matrix
lda_cm = confusion_matrix(y, y_pred_lda)
```

[81]:
```
conf_matrix=pd.DataFrame(data=lda_cm,columns=['Predicted:0','Predicted:
↪1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu")
```

[81]: <AxesSubplot:>



[82]:
```
lda_cm
```

[82]:
```
array([[3131,   46],
       [ 511,   61]], dtype=int64)
```

Train-test split

[93]:
```
from sklearn.model_selection import train_test_split
```

[112]:
```
X_train, X_test, y_train, y_test = train_test_split(X , y, test_size=0.
↪2, random_state = 11)
```

[113]:
```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(2999, 14) (750, 14) (2999,) (750,)
```

[114]:
```
# model
clf_LR =LogisticRegression()
clf_LR.fit(X_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning:␣
↪lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown␣
↪in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[114]:
```
LogisticRegression()
```

[115]:
```
# predictions
y_test_pred = clf_LR.predict(X_test)
```

[116]:
```
y_test_pred
```

[116]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0], dtype=int64)
```

Confusion matrix and model accuracy

[117]:
```python
from sklearn.metrics import accuracy_score
```

[118]:
```python
conf_mat = confusion_matrix(y_test,y_test_pred)
```

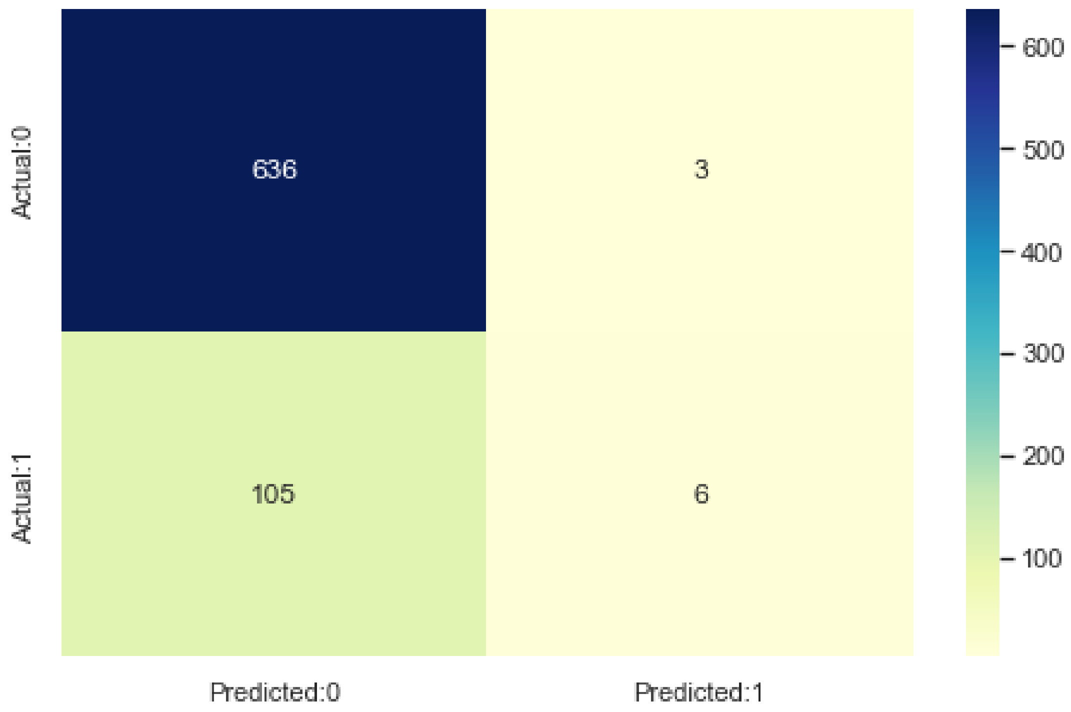
[119]:
```python
accuracy_score(y_test, y_test_pred)
```

[119]: 0.856

[120]:
```python
cm2 = confusion_matrix(y_test, y_test_pred)
```

[121]:
```python
conf_matrix=pd.DataFrame(data=cm2,columns=['Predicted:0','Predicted:
 ↪1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu")
```

[121]: <AxesSubplot:>



[122]:
```
TN=cm2[0,0]
TP=cm2[1,1]
FN=cm2[1,0]
FP=cm2[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
```

[123]:
```
# Model Evaluation - Statistics

print('The acuuracy of the model = TP+TN/(TP+TN+FP+FN) = ',(TP+TN)/
 ↪float(TP+TN+FP+FN),'\n',

'The Missclassification = 1-Accuracy = ',1-((TP+TN)/
 ↪float(TP+TN+FP+FN)),'\n',

'Sensitivity or True Positive Rate = TP/(TP+FN) = ',TP/
 ↪float(TP+FN),'\n',

'Specificity or True Negative Rate = TN/(TN+FP) = ',TN/
 ↪float(TN+FP),'\n',

'Positive Predictive value = TP/(TP+FP) = ',TP/float(TP+FP),'\n',

'Negative predictive Value = TN/(TN+FN) = ',TN/float(TN+FN),'\n',
```

```
'Positive Likelihood Ratio = Sensitivity/(1-Specificity) =␣
↪',sensitivity/(1-specificity),'\n',

'Negative likelihood Ratio = (1-Sensitivity)/Specificity =␣
↪',(1-sensitivity)/specificity)
```

```
The acuuracy of the model = TP+TN/(TP+TN+FP+FN) =  0.856
 The Missclassification = 1-Accuracy =  0.14400000000000002
 Sensitivity or True Positive Rate = TP/(TP+FN) =  0.05405405405405406
 Specificity or True Negative Rate = TN/(TN+FP) =  0.9953051643192489
 Positive Predictive value = TP/(TP+FP) =  0.6666666666666666
 Negative predictive Value = TN/(TN+FN) =  0.8582995951417004
 Positive Likelihood Ratio = Sensitivity/(1-Specificity) =  11.
↪51351351351362
 Negative likelihood Ratio = (1-Sensitivity)/Specificity =  0.
↪9504079551249363
```

From the above statistics, it is clear that the model is more specific than sensitive. That is, negative values are predicted more accurately than positive ones.Finally, we conclude that the attributes that most influence the prediction of heart disease are: male (sex), age, cigsPerDay, totChol, sysBP, and glucose. Since all these show P values lower than 4

With this we can see that apparently men are more susceptible to heart disease, also, we could see that while the subject has a greater age, he will be more prone to suffer from said disease.

This model had an accuracy of 85.6%, being more specific than sensitive.