Untitled

September 9, 2024

```
[337]: import pandas as pd
       import numpy as np
       import matplotlib.pyplot as plt
       import seaborn as sns
       pd.options.display.float_format='{:,.2f}'.format # to round all number to 0.2
       sns.set()
```

About Dataset

Column Name

Description

sex

This column represents the gender of the individuals (female-male).

age

This column represents the age of the individuals in the dataset. Age is a crucial factor in assessing the risk of coronary heart disease.

education

This column represents the level of education of the individuals. It could be coded using categorical values indicating different levels of education attainment.

smokingStatus

This column likely represents the smoking status of the individuals, indicating whether they are smokers (yes) or non-smokers (no).

cigsPerDay

If an individual is a smoker, this column represents the number of cigarettes smoked per day.

BPMeds

This column indicates whether the individual is taking blood pressure medications (binary: 0 for not taking, 1 for taking).

prevalentStroke

This column indicates whether an individual has had a stroke prior to the study (binary: 0 for no, 1 for yes).

prevalentHyp

This column indicates whether an individual has hypertension (binary: 0 for no, 1 for yes).

diabetes

This column indicates whether an individual has diabetes (binary: 0 for no, 1 for yes).

totChol

This column represents the total cholesterol level of the individuals.

sysBP

This column represents the systolic blood pressure of the individuals.

diaBP

This column represents the diastolic blood pressure of the individuals.

BMI

This column represents the Body Mass Index (BMI) of the individuals, which is a measure of body fat based on height and weight.

heartRate

This column represents the resting heart rate of the individuals.

glucose

This column represents the fasting blood glucose level of the individuals.

CHDRisk

This column likely represents the Ten-Year Coronary Heart Disease (CHD) Risk for each individual, which is the target variable that you may want to predict or analyze.

[359]:
```
data=pd.read_csv('Heart_Disease.csv')
data.head()
```

[359]:
```
      sex  age  education smokingStatus  cigsPerDay  BPMeds  prevalentStroke  \
0    male   39          4            no           0       0                0
1  female   46          2            no           0       0                0
2    male   48          1           yes          20       0                0
3  female   61          3           yes          30       0                0
4  female   46          3           yes          23       0                0

   prevalentHyp diabetes  totChol  sysBP  diaBP   BMI  heartRate  glucose  \
0             0       no      195 106.00  70.00 26.97         80       77
1             0       no      250 121.00  81.00 28.73         95       76
2             0       no      245 127.50  80.00 25.34         75       70
3             1       no      225 150.00  95.00 28.58         65      103
4             0       no      285 130.00  84.00 23.10         85       85

  CHDRisk
0      no
```

```
1      no
2      no
3     yes
4      no
```

1 Data_Size

```
[360]: print(f'shape of data is {df.shape[0]} , {df.shape[1]}')
```

```
shape of data is 3674 , 16
```

```
[361]: df=data.copy()
```

2 Data_Types

```
[362]: df.dtypes
```

```
[362]: sex                  object
        age                   int64
        education             int64
        smokingStatus        object
        cigsPerDay            int64
        BPMeds                int64
        prevalentStroke       int64
        prevalentHyp          int64
        diabetes             object
        totChol               int64
        sysBP               float64
        diaBP               float64
        BMI                 float64
        heartRate             int64
        glucose               int64
        CHDRisk              object
        dtype: object
```

```
[363]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3674 entries, 0 to 3673
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   sex              3663 non-null   object
 1   age              3674 non-null   int64
 2   education        3674 non-null   int64
 3   smokingStatus    3661 non-null   object
 4   cigsPerDay       3674 non-null   int64
 5   BPMeds           3674 non-null   int64
 6   prevalentStroke  3674 non-null   int64
 7   prevalentHyp     3674 non-null   int64
 8   diabetes         3674 non-null   object
 9   totChol          3674 non-null   int64
 10  sysBP            3674 non-null   float64
 11  diaBP            3674 non-null   float64
 12  BMI              3674 non-null   float64
 13  heartRate        3674 non-null   int64
 14  glucose          3674 non-null   int64
 15  CHDRisk          3674 non-null   object
dtypes: float64(3), int64(9), object(4)
memory usage: 459.4+ KB
```

3 Unique Values

[364]:
```
for col in df.columns:
    print(f' {col}:  \n number of unique value for each column {df[col].
 ↪nunique()} , \n unique values is {df[col].unique()}')
    print('='*100)
```

```
 sex:
 number of unique value for each column 2 ,
 unique values is ['male' 'female' nan]
================================================================================
====================
 age:
 number of unique value for each column 39 ,
 unique values is [39 46 48 61 43 63 45 52 50 41 38 42 44 47 35 60 36 59 54 37
53 49 65 51
 62 40 56 67 57 66 64 55 58 68 34 33 32 70 69]
================================================================================
====================
 education:
 number of unique value for each column 4 ,
 unique values is [4 2 1 3]
================================================================================
====================
 smokingStatus:
 number of unique value for each column 2 ,
 unique values is ['no' 'yes' nan]
================================================================================
====================
 cigsPerDay:
 number of unique value for each column 33 ,
 unique values is [ 0 20 30 23 15 10  5 35 43  1 40  3  9  2 12  4 18 60 25 45
8 13 11  7
```

```
  6 38 50 29 17 16 19 70 14]
================================================================================
====================
 BPMeds:
 number of unique value for each column 2 ,
 unique values is [0 1]
================================================================================
====================
 prevalentStroke:
 number of unique value for each column 2 ,
 unique values is [0 1]
================================================================================
====================
 prevalentHyp:
 number of unique value for each column 2 ,
 unique values is [0 1]
================================================================================
====================
 diabetes:
 number of unique value for each column 2 ,
 unique values is ['no' 'yes']
================================================================================
====================
 totChol:
 number of unique value for each column 241 ,
 unique values is [195 250 245 225 285 228 205 313 260 254 247 294 332 221 232
291 190 234
 215 270 272 295 226 209 214 178 233 180 243 237 311 208 252 261 179 267
 216 240 266 255 185 220 235 212 223 300 302 175 189 258 202 183 274 170
 210 197 326 188 256 244 193 239 296 269 275 268 265 173 273 290 278 264
 282 257 241 288 200 213 303 246 150 187 286 154 279 293 259 219 230 320
 312 165 159 174 242 301 167 308 325 229 236 224 253 464 248 171 186 227
 249 176 196 310 164 135 238 207 342 287 182 352 284 203 262 155 323 206
 283 319 194 340 328 222 368 218 276 339 231 198 201 277 304 177 199 292
 305 152 161 168 181 251 271 217 370 439 145 263 330 157 398 162 314 166
 160 281 289 355 307 156 329 143 211 298 334 192 184 204 280 191 163 318
 353 360 335 158 346 169 140 324 600 315 392 322 306 309 149 137 172 317
 358 345 391 410 297 338 148 372 366 333 327 344 144 390 321 405 359 350
 336 380 299 124 371 113 354 382 364 341 133 367 153 432 351 337 363 331
 316 361 453 347 373 385 119]
================================================================================
====================
 sysBP:
 number of unique value for each column 231 ,
 unique values is [106.  121.  127.5 150.  130.  180.  138.  100.  141.5 162.
133.  131.
 142.  124.  140.  112.  122.  139.  108.  148.  132.  137.5 102.  182.
 115.  147.  124.5 160.  153.  111.  116.5 206.   96.  179.5 119.  116.
 156.5 145.  114.  143.5 158.  157.  123.5 126.5 136.  154.  190.  107.
 112.5 110.  138.5 155.  151.  152.  179.  113.  200.  132.5 126.  123.
 134.  141.  135.  187.  127.  160.5 105.  109.  128.  118.  117.5 149.
 180.5 136.5 212.  191.  121.5 173.  144.  129.5 117.  125.  144.5 170.
 137.   94.  166.  177.5 129.  159.  130.5 107.5 189.  168.  197.5 146.
 174.   98.  131.5 101.  158.5  97.  151.5  97.5 120.  204.  157.5 140.5
 171.  215.   95.  156.  122.5 178.  146.5 113.5 197.   90.  109.5 165.
  95.5 209.  162.5 295.  103.  134.5 115.5 174.5 163.  118.5 185.  220.
 164.  120.5  98.5 161.  139.5 168.5 176.  163.5 128.5 167.  205.5 119.5
 167.5 152.5 186.  183.  153.5 147.5 175.  142.5 192.   96.5 159.5 177.
 102.5 244.  104.  213.  199.  184.  198.  114.5 125.5 111.5 105.5 143.
 161.5 164.5 171.5 108.5 201.  148.5 172.  243.  145.5 187.5  99.  181.
 133.5 100.5 135.5 172.5 103.5 149.5 182.5 186.5 217.  196.  193.  110.5
 155.5  92.  169.  166.5 202.  150.5 195.  232.   85.5 184.5 188.  205.
 169.5 210.  181.5 188.5 176.5  92.5 202.5 154.5  83.5 106.5 170.5  93.
 175.5 207.5 199.5 101.5 248.   99.5  85.  230.  214.  192.5 104.5 194.
  93.5 207.  185.5]
================================================================================
====================
 diaBP:
 number of unique value for each column 142 ,
 unique values is [ 70.   81.   80.   95.   84.  110.   71.   89.  107.   76.
88.   94.
  90.   78.   84.5  70.5  82.   68.   91.  121.   85.5  85.   74.   92.5
  98.  101.   73.   83.5  92.   63.  114.   77.5  69.   66.   82.5 102.
  79.   75.   87.   99.   60.   67.5  72.5 106.   86.5 104.   86.   61.5
  71.5  76.5  64.   77.   88.5 105.   96.   97.  100.  106.5  93.   80.5
 124.5  61.   83.   67.   74.5  66.5  65.   72.   99.5 122.5  57.   57.5
 111.   78.5 104.5  89.5 112.   55.  120.  118.   59.  133.   95.5  96.5
 135.   64.5  68.5  98.5  62.  117.   59.5 103.   75.5  73.5  69.5  87.5
 108.   93.5  90.5 114.5  62.5  94.5 140.  124.   91.5 115.  109.  102.5
  65.5 105.5 103.5  63.5  79.5 107.5 142.5 109.5  58.   97.5 116.5 100.5
 116.  119.   81.5  54.  132.  101.5 136.   51.  128.  125.  130.  110.5
 113.   53.  108.5 112.5  52.   48.   56.   60.5 115.5 127.5]
================================================================================
====================
 BMI:
 number of unique value for each column 1297 ,
 unique values is [26.97 28.73 25.34 … 26.7  43.67 19.71]
================================================================================
====================
 heartRate:
 number of unique value for each column 72 ,
 unique values is [ 80  95  75  65  85  77  60  79  76  93  72  98  64  70  71
62  73  90
  96  68  63  88  78  83 100  84  57  50  74  86  55  92  66  87 110  81
  56  89  82  54  69  67  52  61 140 130  58 104  94 105  91  53 108 106
  59 107  48 112 125 103  44  47  45  97 122 102 120  99 115 143 101  46]
```

```
================================================================================
====================
 glucose:
 number of unique value for each column 138 ,
 unique values is [ 77  76  70 103  85  99  78  79  88  61  64  84  72  89  65
113  75  83
  66  74  63  87 225  90  80 100 215  98  95  94  55  82  93  73  45 202
  68  97 104  96 126 120 105  71  56  60 117  62 102  58  92 109  86 107
  54  67  69  57  91 132 150  59  81 115 140 112 118 114 160 110 123 108
 145 122 137 106 127 205 130 101  47  53 216 163 144 116 121 172 124 111
  40 186 223 325  44 156 268  50 274 292 255 136 206 131 148  43 173 386
 155 147 170  52 320 254 394 270 244 183 142 119 167 135 207 129 177 250
 294 125 332 368 348 370 193 191 256 235 210 260]
================================================================================
====================
 CHDRisk:
 number of unique value for each column 2 ,
 unique values is ['no' 'yes']
================================================================================
====================
```

Categorizing Age into Age Groups

[365]:
```
df['age_group'] = pd.cut(df['age'], bins=[0, 30, 40, 60, 80], labels=['0-30',␣
↪'30-40', '40-60', '60-80'])
df.drop(columns='age',inplace=True)
```

Converting Binary Medical Columns to Categorical Values for easier interpretation and analysis

[366]:
```
df['BPMeds']=df['BPMeds'].astype('object')
df['prevalentStroke']=df['prevalentStroke'].astype('object')
df['prevalentHyp']=df['prevalentHyp'].astype('object')

df['BPMeds']=df['BPMeds'].replace([0,1],['no','yes'])
df['prevalentStroke']=df['prevalentStroke'].replace([0,1],['no','yes'])
df['prevalentHyp']=df['prevalentHyp'].replace([0,1],['no','yes'])
```

4 Show Data

[367]:
```
df.head()
```

[367]:
```
      sex  education smokingStatus  cigsPerDay BPMeds prevalentStroke  \
0    male          4            no           0     no              no
1  female          2            no           0     no              no
2    male          1           yes          20     no              no
3  female          3           yes          30     no              no
4  female          3           yes          23     no              no
```

```
  prevalentHyp diabetes  totChol  sysBP  diaBP   BMI  heartRate  glucose  \
0           no       no      195 106.00  70.00 26.97         80       77
1           no       no      250 121.00  81.00 28.73         95       76
2           no       no      245 127.50  80.00 25.34         75       70
3          yes       no      225 150.00  95.00 28.58         65      103
4           no       no      285 130.00  84.00 23.10         85       85

  CHDRisk age_group
0      no     30-40
1      no     40-60
2      no     40-60
3     yes     60-80
4      no     40-60
```

[368]:
```
df.tail()
```

[368]:
```
       sex  education smokingStatus  cigsPerDay BPMeds prevalentStroke  \
3669  male          3           yes          25     no              no
3670  male          3           yes          25     no              no
3671  male          3           yes          25     no              no
3672  male          3           yes          25     no              no
3673  male          2           yes          25     no              no

     prevalentHyp diabetes  totChol  sysBP  diaBP   BMI  heartRate  glucose  \
3669           no       no      208 137.50  82.50 25.58         75       63
3670           no       no      208 137.50  82.50 25.58         75       63
3671           no       no      208 137.50  82.50 25.58         75       63
3672           no       no      208 137.50  82.50 25.58         75       63
3673           no       no      208 137.50  82.50 25.97         69       68

     CHDRisk age_group
3669     yes     40-60
3670     yes     40-60
3671     yes     40-60
3672     yes     40-60
3673     yes     40-60
```

[369]:
```
df.sample(2)
```

[369]:
```
         sex  education smokingStatus  cigsPerDay BPMeds prevalentStroke  \
1573    male          1           yes          35     no              no
1059  female          2           yes          20     no              no

     prevalentHyp diabetes  totChol  sysBP  diaBP   BMI  heartRate  glucose  \
1573           no       no      188 120.00  82.50 31.67         80       68
1059           no       no      149 122.00  72.00 21.30         85       75
```

```
      CHDRisk age_group
1573      no     40-60
1059      no     30-40
```

5 Missing_Values

[370]:
```
df.isna().sum()
```

[370]:
```
sex                11
education           0
smokingStatus      13
cigsPerDay          0
BPMeds              0
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol             0
sysBP               0
diaBP               0
BMI                 0
heartRate           0
glucose             0
CHDRisk             0
age_group           0
dtype: int64
```

[371]:
```
df=df.dropna()
```

[372]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3652 entries, 0 to 3673
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   sex              3652 non-null   object
 1   education        3652 non-null   int64
 2   smokingStatus    3652 non-null   object
 3   cigsPerDay       3652 non-null   int64
 4   BPMeds           3652 non-null   object
 5   prevalentStroke  3652 non-null   object
 6   prevalentHyp     3652 non-null   object
 7   diabetes         3652 non-null   object
 8   totChol          3652 non-null   int64
 9   sysBP            3652 non-null   float64
 10  diaBP            3652 non-null   float64
 11  BMI              3652 non-null   float64
 12  heartRate        3652 non-null   int64
 13  glucose          3652 non-null   int64
 14  CHDRisk          3652 non-null   object
 15  age_group        3652 non-null   category
dtypes: category(1), float64(3), int64(5), object(7)
memory usage: 460.3+ KB
```

6 Duplicated_Values

[374]:
```
duplicated_data=df[df.duplicated(keep=False)]
duplicated_data
```

[374]:
```
       sex  education smokingStatus  cigsPerDay BPMeds prevalentStroke  \
3118  male          3           yes          25     no              no
3658  male          3           yes          25     no              no
3659  male          3           yes          25     no              no
3660  male          3           yes          25     no              no
3661  male          3           yes          25     no              no
3662  male          3           yes          25     no              no
3663  male          3           yes          25     no              no
3664  male          3           yes          25     no              no
3665  male          3           yes          25     no              no
3666  male          3           yes          25     no              no
3667  male          3           yes          25     no              no
3668  male          3           yes          25     no              no
3669  male          3           yes          25     no              no
3670  male          3           yes          25     no              no
3671  male          3           yes          25     no              no
3672  male          3           yes          25     no              no

     prevalentHyp diabetes  totChol  sysBP  diaBP   BMI  heartRate  glucose  \
3118           no       no      208 137.50  82.50 25.58         75       63
3658           no       no      208 137.50  82.50 25.58         75       63
3659           no       no      208 137.50  82.50 25.58         75       63
3660           no       no      208 137.50  82.50 25.58         75       63
3661           no       no      208 137.50  82.50 25.58         75       63
3662           no       no      208 137.50  82.50 25.58         75       63
3663           no       no      208 137.50  82.50 25.58         75       63
3664           no       no      208 137.50  82.50 25.58         75       63
3665           no       no      208 137.50  82.50 25.58         75       63
3666           no       no      208 137.50  82.50 25.58         75       63
3667           no       no      208 137.50  82.50 25.58         75       63
3668           no       no      208 137.50  82.50 25.58         75       63
3669           no       no      208 137.50  82.50 25.58         75       63
3670           no       no      208 137.50  82.50 25.58         75       63
3671           no       no      208 137.50  82.50 25.58         75       63
3672           no       no      208 137.50  82.50 25.58         75       63
```

```
     CHDRisk age_group
3118     yes     40-60
3658     yes     40-60
3659     yes     40-60
3660     yes     40-60
3661     yes     40-60
3662     yes     40-60
3663     yes     40-60
3664     yes     40-60
3665     yes     40-60
3666     yes     40-60
3667     yes     40-60
3668     yes     40-60
3669     yes     40-60
3670     yes     40-60
3671     yes     40-60
3672     yes     40-60
```

[379]:
```
df.duplicated().sum()
```

[379]:
```
0
```

[380]:
```
df.drop_duplicates(inplace=True)
```

[381]:
```
df.duplicated().any().sum()
```

[381]:
```
0
```

[382]:
```
df.isna().sum()
```

[382]:
```
sex                0
education          0
smokingStatus      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
CHDRisk            0
age_group          0
```

```
dtype: int64
```

7 Statistical_OverView

[384]:
```
df.describe().T
```

[384]:
```
                count   mean   std    min    25%    50%    75%    max
education  3,637.00   1.98  1.02   1.00   1.00   2.00   3.00   4.00
cigsPerDay 3,637.00   9.03 11.91   0.00   0.00   0.00  20.00  70.00
totChol    3,637.00 236.88 44.13 113.00 206.00 234.00 263.00 600.00
sysBP      3,637.00 132.36 22.08  83.50 117.00 128.00 144.00 295.00
diaBP      3,637.00  82.90 11.96  48.00  75.00  82.00  90.00 142.50
BMI        3,637.00  25.79  4.06  15.54  23.08  25.38  28.04  56.80
heartRate  3,637.00  75.75 11.99  44.00  68.00  75.00  82.00 143.00
glucose    3,637.00  81.81 23.77  40.00  71.00  78.00  87.00 394.00
```

8 Analysis

[385]:
```
df.columns
```

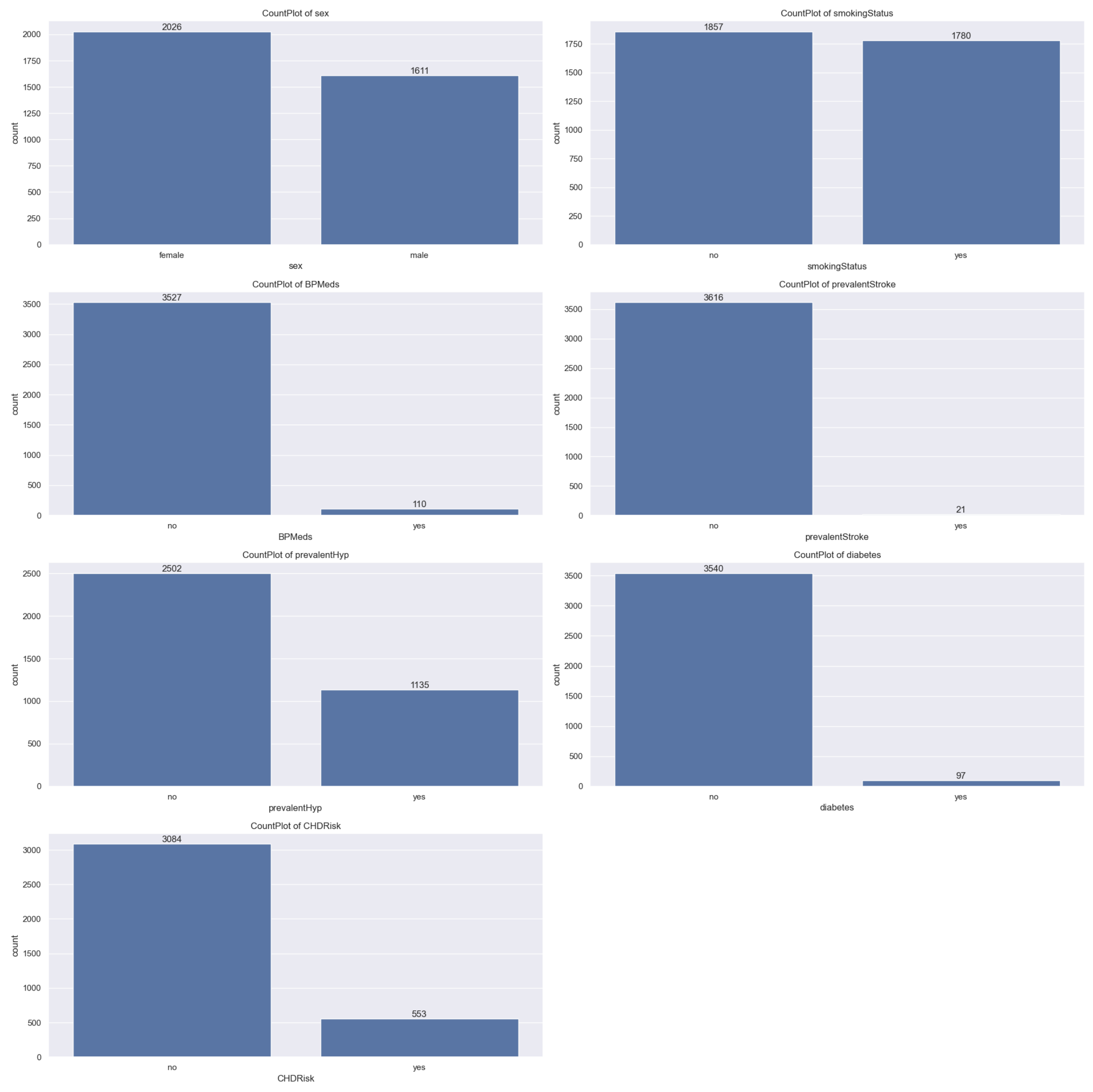
[385]:
```
Index(['sex', 'education', 'smokingStatus', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'CHDRisk', 'age_group'],
      dtype='object')
```

1- univariate

Count Plots for Categorical Variables

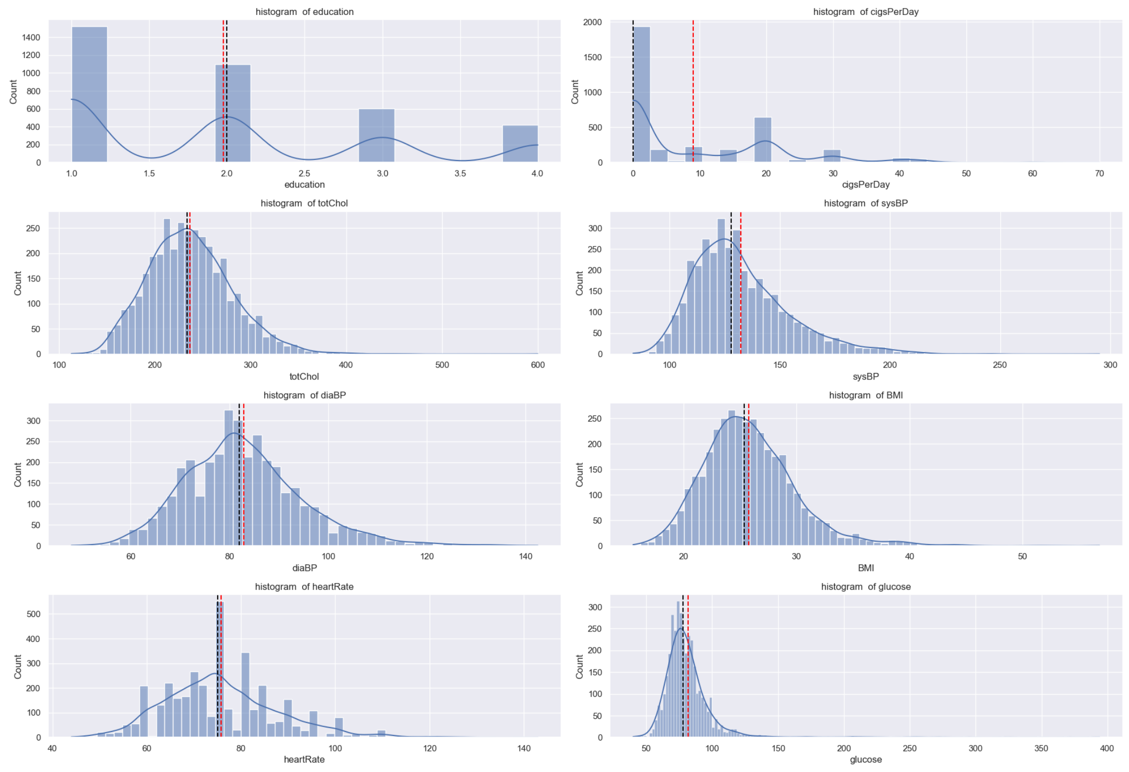
[263]:
```
plt.figure(figsize=(20,20))
for i , col in enumerate(df.select_dtypes('object')):
    plt.subplot(4,2,i+1)
    ax=sns.countplot(x=df[col],order=df[col].value_counts().index)
    ax.bar_label(ax.containers[0])# this line to show count for each col
    plt.title(f'CountPlot of {col}')
    plt.tight_layout()
    print(f'Counts for {col}:\n', df[col].value_counts(), '\n')

plt.show()
```

```
Counts for sex:
 sex
female    2026
male      1611
Name: count, dtype: int64

Counts for smokingStatus:
 smokingStatus
no     1857
yes    1780
Name: count, dtype: int64

Counts for BPMeds:
 BPMeds
no     3527
yes     110
Name: count, dtype: int64

Counts for prevalentStroke:
 prevalentStroke
no     3616
yes      21
Name: count, dtype: int64

Counts for prevalentHyp:
 prevalentHyp
no     2502
yes    1135
Name: count, dtype: int64

Counts for diabetes:
 diabetes
no     3540
yes      97
Name: count, dtype: int64

Counts for CHDRisk:
 CHDRisk
no     3084
yes     553
Name: count, dtype: int64
```



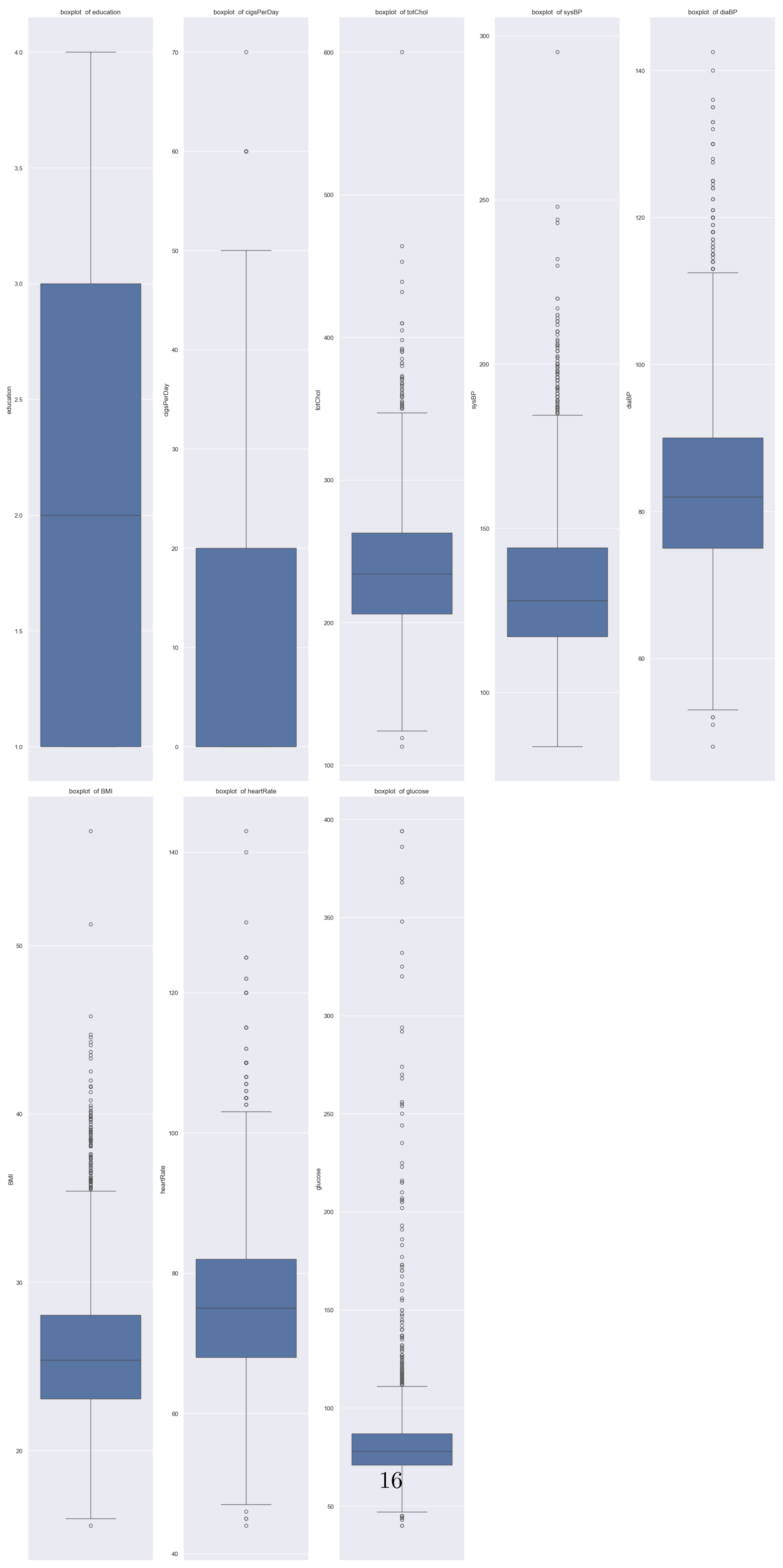
Box Plots for numerical columns

[386]:
```
plt.figure(figsize=(20,20))
for i , col in enumerate(df.select_dtypes('number')):
    plt.subplot(6,2,i+1)
    ax=sns.histplot(df[col],kde=True)
    ax.axvline(df[col].mean(),color='red',linestyle='--')
    ax.axvline(df[col].median(),color='black',linestyle='--')
    plt.title(f'histogram  of {col}')
    plt.tight_layout()
plt.show()
```

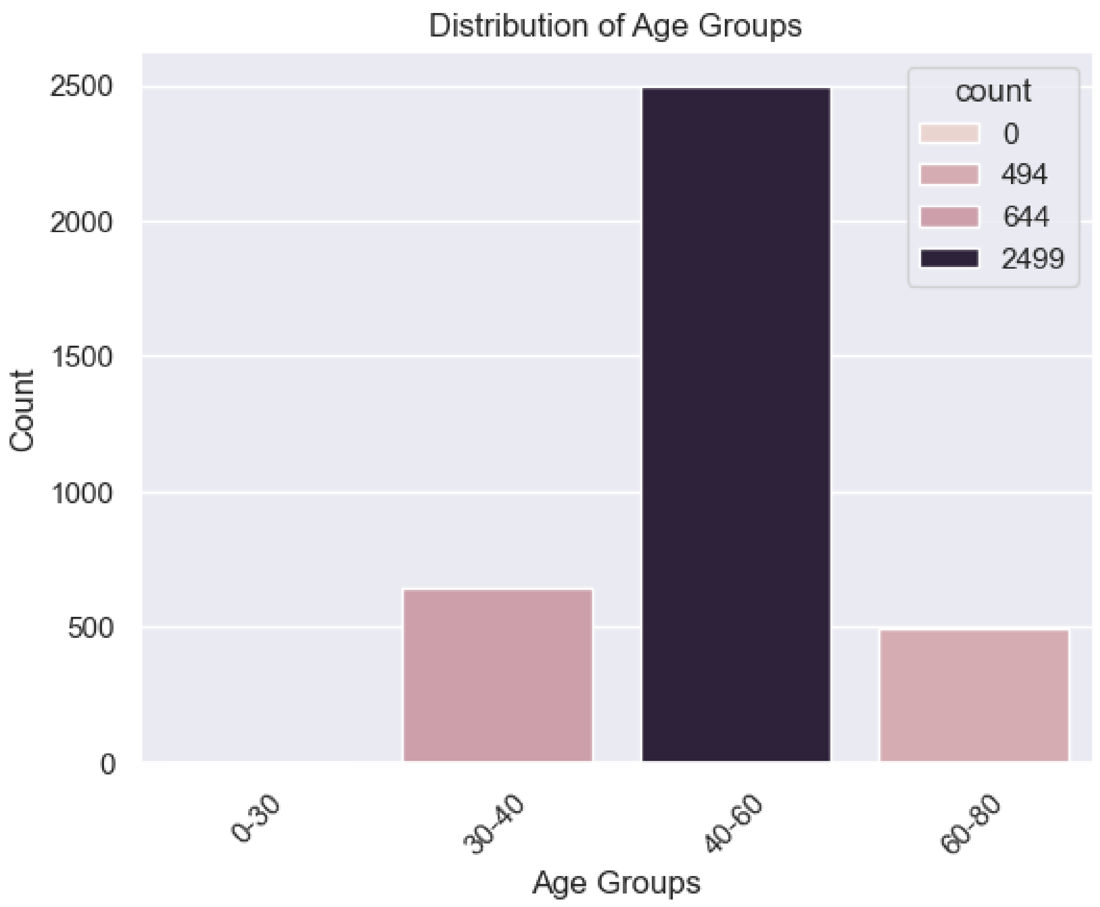


data is approximately normally distributed bec mean is closer to Median in histogram and descriptive stat

[387]:
```
plt.figure(figsize=(20,40))
for i , col in enumerate(df.select_dtypes('number')):
    plt.subplot(2,5,i+1)
    ax=sns.boxplot(df[col])
    plt.title(f'boxplot  of {col}')
    plt.tight_layout()
plt.show()
```


boxplot of education
education
1.0
1.5
2.0
2.5
3.0
3.5
4.0
boxplot of cigsPerDay
cigsPerDay
0
10
20
30
40
50
60
70
boxplot of totChol
totChol
100
200
300
400
500
600
boxplot of sysBP
sysBP
100
150
200
250
300
boxplot of diaBP
diaBP
60
80
100
120
140
boxplot of BMI
BMI
20
30
40
50
boxplot of heartRate
heartRate
40
60
80
100
120
140
boxplot of glucose
glucose
50
100
150
200
250
300
350
400

[388]:
```
age_grouped_valueCount=df['age_group'].value_counts()
# age_grouped_valueCount
sns.barplot(x=age_grouped_valueCount.index,y=age_grouped_valueCount.
 ↪values,hue=age_grouped_valueCount)
plt.title('Distribution of Age Groups')
plt.xlabel('Age Groups')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



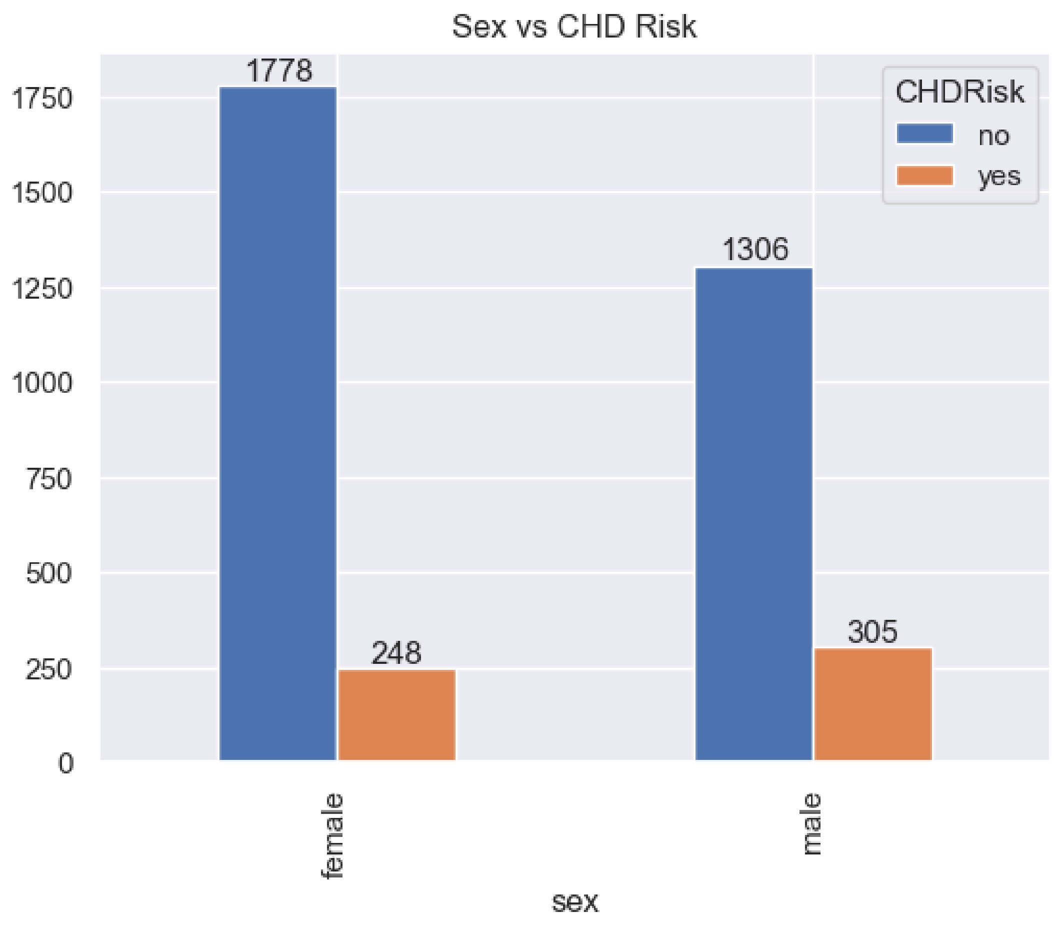
2- Bivariate

a. Categorical vs Categorical:

[389]:
```
crossTab_df=pd.crosstab(df['sex'],df['CHDRisk'])
print(crossTab_df)
```

```
ax=crossTab_df.plot(kind='bar',title='Sex vs CHD Risk')
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])

plt.show()
```

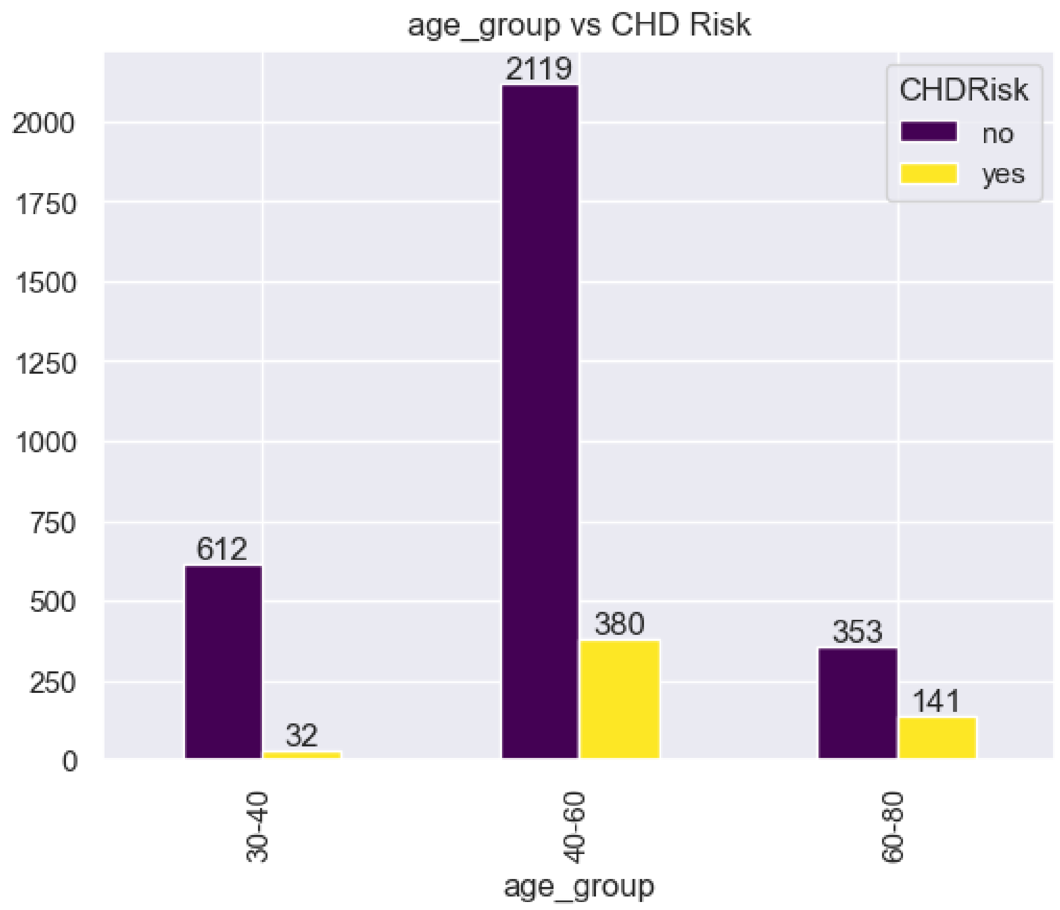
```
CHDRisk    no  yes
sex
female   1778  248
male     1306  305
```



[390]:
```
crossTab_df=pd.crosstab(df['age_group'],df['CHDRisk'])
print(crossTab_df)

ax=crossTab_df.plot(kind='bar',title='age_group vs CHD Risk',colormap='viridis')
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
```

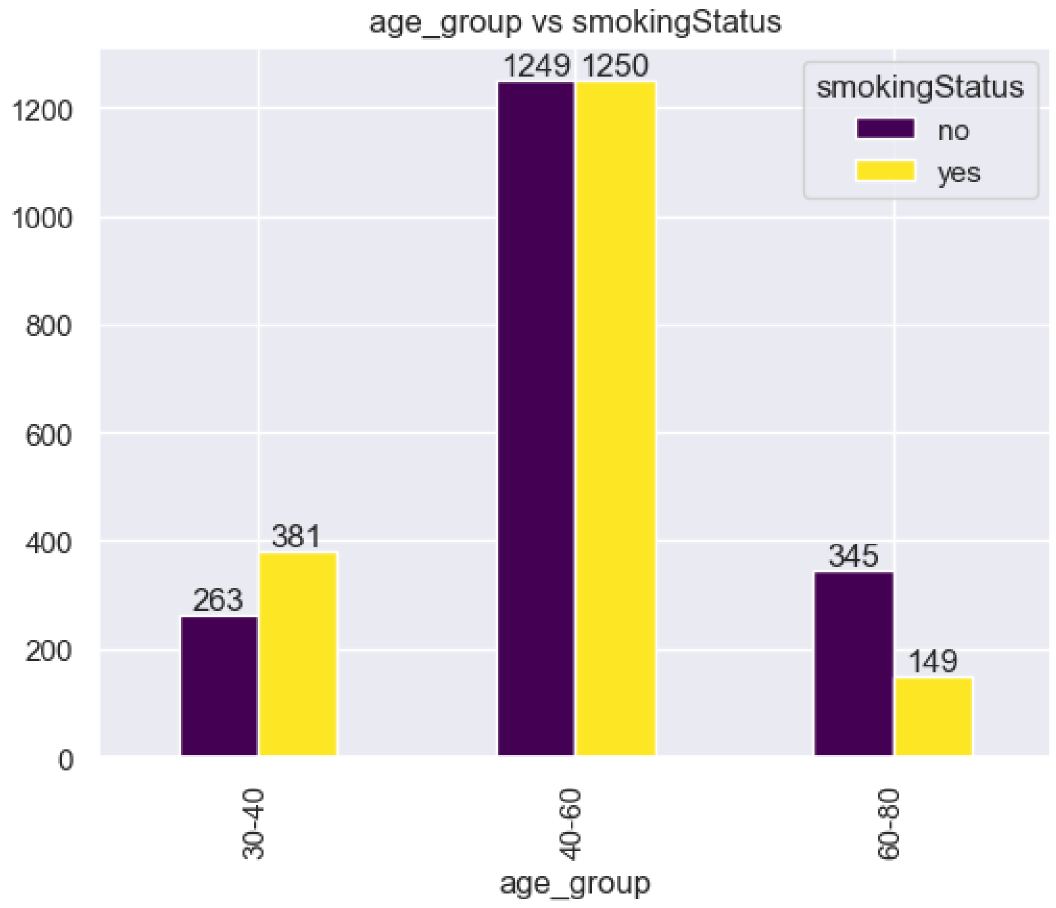
```
plt.show()
```

```
CHDRisk       no  yes
age_group
30-40        612   32
40-60       2119  380
60-80        353  141
```



```
[391]: crossTab_df=pd.crosstab(df['age_group'],df['smokingStatus'])
print(crossTab_df)

ax=crossTab_df.plot(kind='bar',title='age_group vs␣
 ↪smokingStatus',colormap='viridis')
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])

plt.show()
```

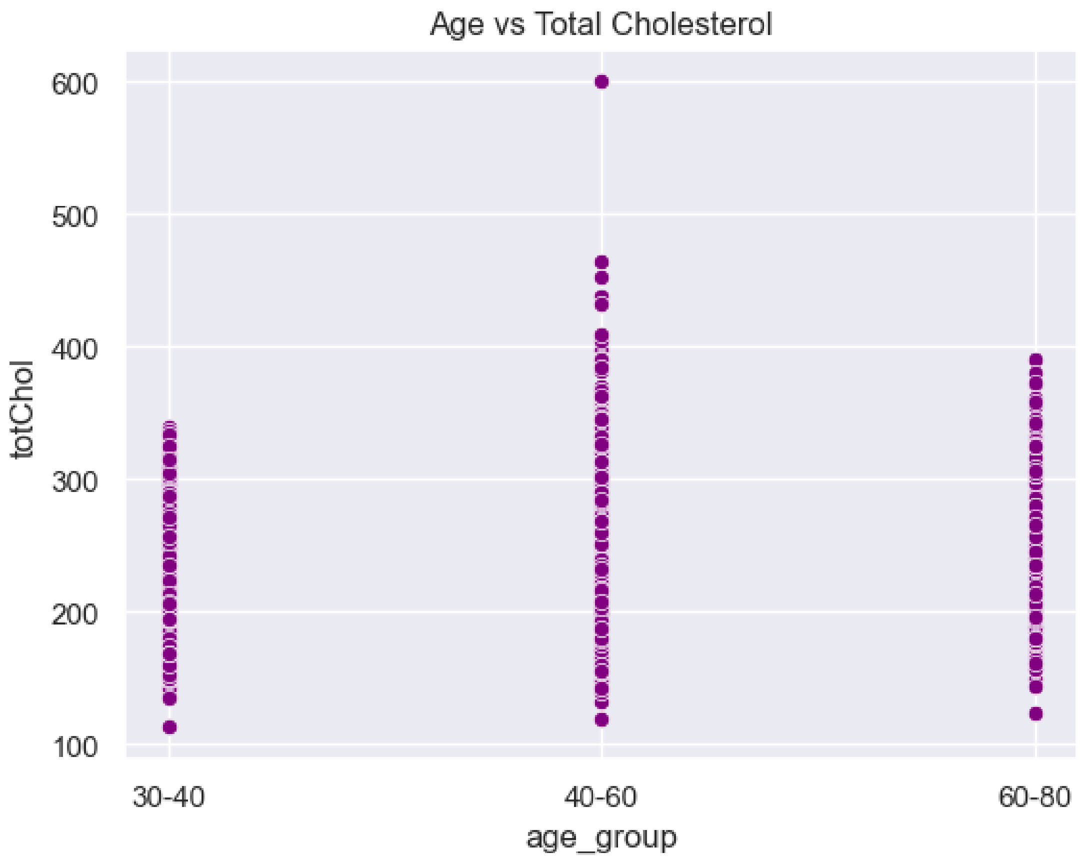
```
smokingStatus     no   yes
age_group
30-40            263   381
40-60           1249  1250
60-80            345   149
```



b. Numerical vs Categorical

```
[392]: sns.scatterplot(data=df,x='age_group',y='totChol',color='purple')
       plt.title('Age vs Total Cholesterol')
```

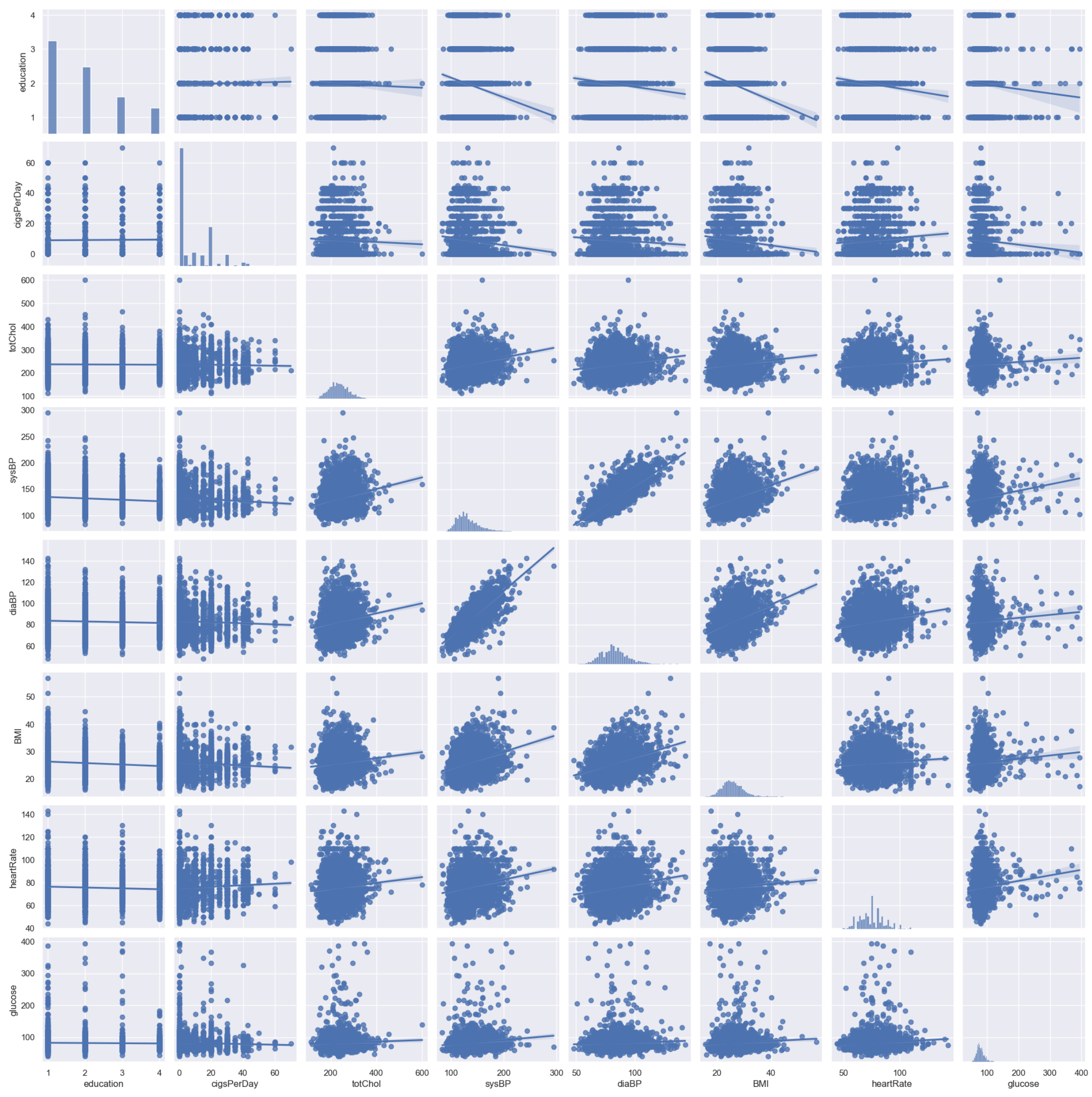
```
[392]: Text(0.5, 1.0, 'Age vs Total Cholesterol')
```



3- Multivariate Analysis

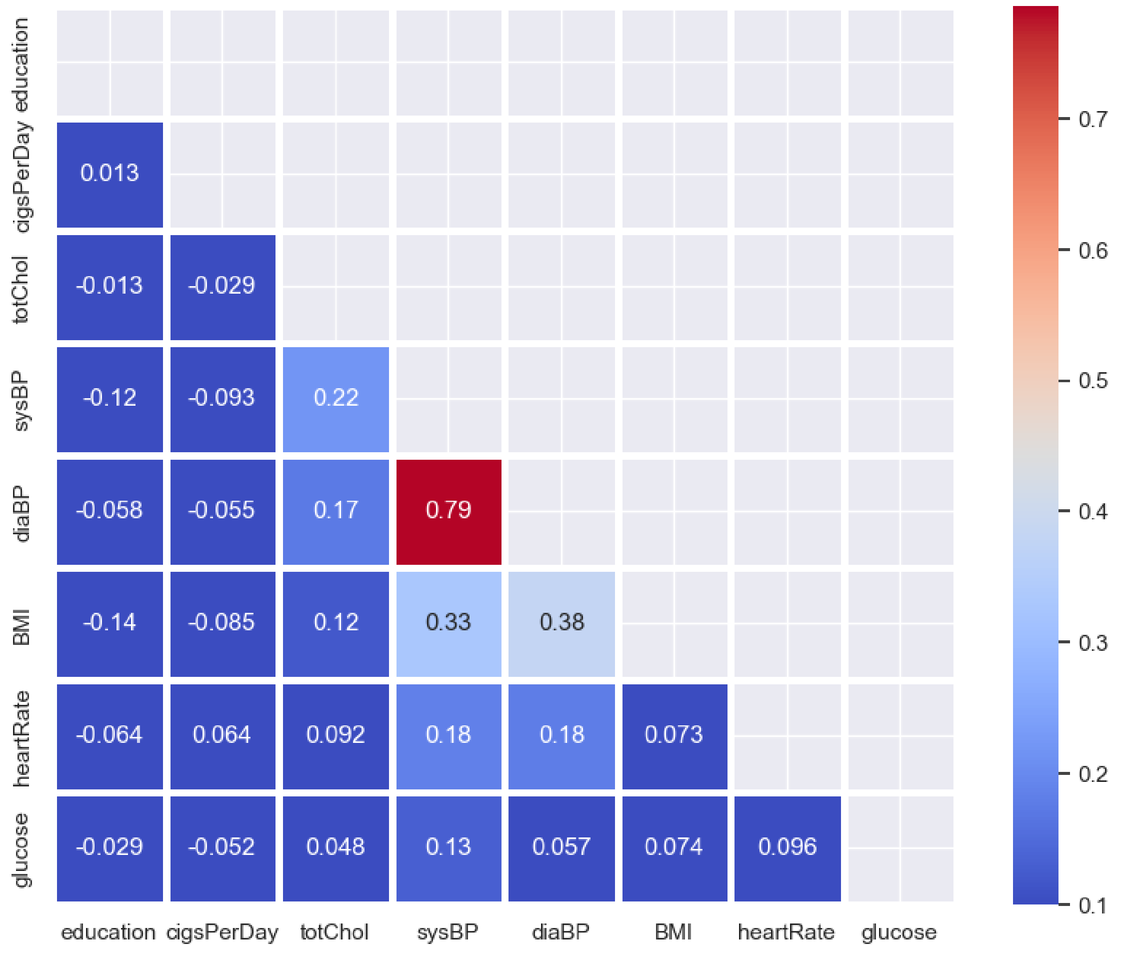
```
[393]: numerical_columns=df.select_dtypes('number')
       sns.pairplot(data=numerical_columns,kind='reg')
```

```
[393]: <seaborn.axisgrid.PairGrid at 0x242ab284fb0>
```



[394]:
```
# Generate a mask for the upper triangle
mask = np.zeros_like(numerical_columns.corr())
mask[np.triu_indices_from(mask)] = True

plt.figure(figsize=(10, 8))

sns.heatmap(numerical_columns.corr(),
            annot=True,
            vmin=0.1,            # Set the minimum correlation value color
            mask=mask,           # Apply the mask to hide the upper triangle
            linewidths=3.5,      # Increase the space between squares
            linecolor='white',   # Set the line color between the squares
            cmap='coolwarm')
```

```
plt.show()
```



there is a strong correlation between sysBP and diaBp

9 Health Metrics: Normal, Borderline, and High Ranges

Metric	**Normal Range**	**Borderline/Elevated**	**High/Abnormal**
Total Cholesterol (totChol)	Less than 200 mg/dL	200-239 mg/dL	240 mg/dL and above
Systolic Blood Pressure (sysBP)	90-120 mmHg	120-129 mmHg	130 mmHg and above

Metric	Normal Range	Borderline/Elevated	High/Abnormal
Diastolic Blood Pressure (diaBP)	60-80 mmHg	80-89 mmHg	90 mmHg and above
Body Mass Index (BMI)	18.5-24.9	25-29.9	30 and above (Obesity)
Heart Rate (heartRate)	60-100 bpm	-	Less than 60 bpm (Bradycardia), more than 100 bpm (Tachycardia)
Glucose	70-99 mg/dL (fasting)	100-125 mg/dL (Prediabetes)	126 mg/dL and above (Diabetes)

Number of Individuals Classified as Healthy Based on Key Health Metrics

[395]:
```
healty_conditions=(
               (df['sex'].isin(['male','female']))&
               (df['smokingStatus'].isin(['no','yes']))&
               (df['cigsPerDay'].between(0,20))&
               (df['BPMeds']=='no')&
               (df['prevalentStroke']=='no')&
                (df['prevalentHyp']=='no')&
                (df['diabetes']== 'no')&
                (df['diaBP'].between(60,80))&
                (df['sysBP'].between(90,120))&
                (df['BMI'].between(18.5,24.9))&
                (df['heartRate'].between(60,100))&
                (df['glucose'].between(70,99))&
                (df['totChol']<200)
)
healthy_People=df[healty_conditions]
length_healthy_people = len(healthy_People)
print(f"Number of healthy people: {length_healthy_people}")
```

```
Number of healthy people: 100
```

[398]:
```
healthy_People.sample(10)
```

[398]:
```
         sex  education smokingStatus  cigsPerDay BPMeds prevalentStroke  \
530   female          2            no           0     no              no
1469  female          1            no           0     no              no
1729  female          4            no           0     no              no
635   female          1           yes          20     no              no
636   female          3            no           0     no              no
1452  female          1           yes          20     no              no
3524    male          4           yes          20     no              no
1778  female          1           yes          20     no              no
1460  female          2           yes          10     no              no
```

```
1381  female           3            yes            3      no                no

     prevalentHyp diabetes  totChol  sysBP  diaBP   BMI  heartRate  glucose  \
530            no       no      170  98.50  69.50 19.64         71       77
1469           no       no      170 110.00  69.00 23.48         75       83
1729           no       no      150 108.00  70.50 20.42         72       88
635            no       no      168 117.00  74.00 21.51         67       77
636            no       no      197 107.00  73.00 19.78         63       76
1452           no       no      175 117.50  73.50 22.15         65       75
3524           no       no      198 116.00  74.00 23.99         75       78
1778           no       no      166 112.00  73.50 21.64         75       93
1460           no       no      169 119.00  72.00 19.78         60       74
1381           no       no      186 114.00  77.00 21.01         80       85

     CHDRisk age_group
530       no     30-40
1469      no     30-40
1729      no     30-40
635       no     30-40
636       no     40-60
1452      no     30-40
3524      no     40-60
1778     yes     30-40
1460      no     40-60
1381      no     30-40
```

index of 1778 write yes in CHDRisk Although it is among the healthy people

10 Distribution and Classification of Health Metrics and CHDRisk

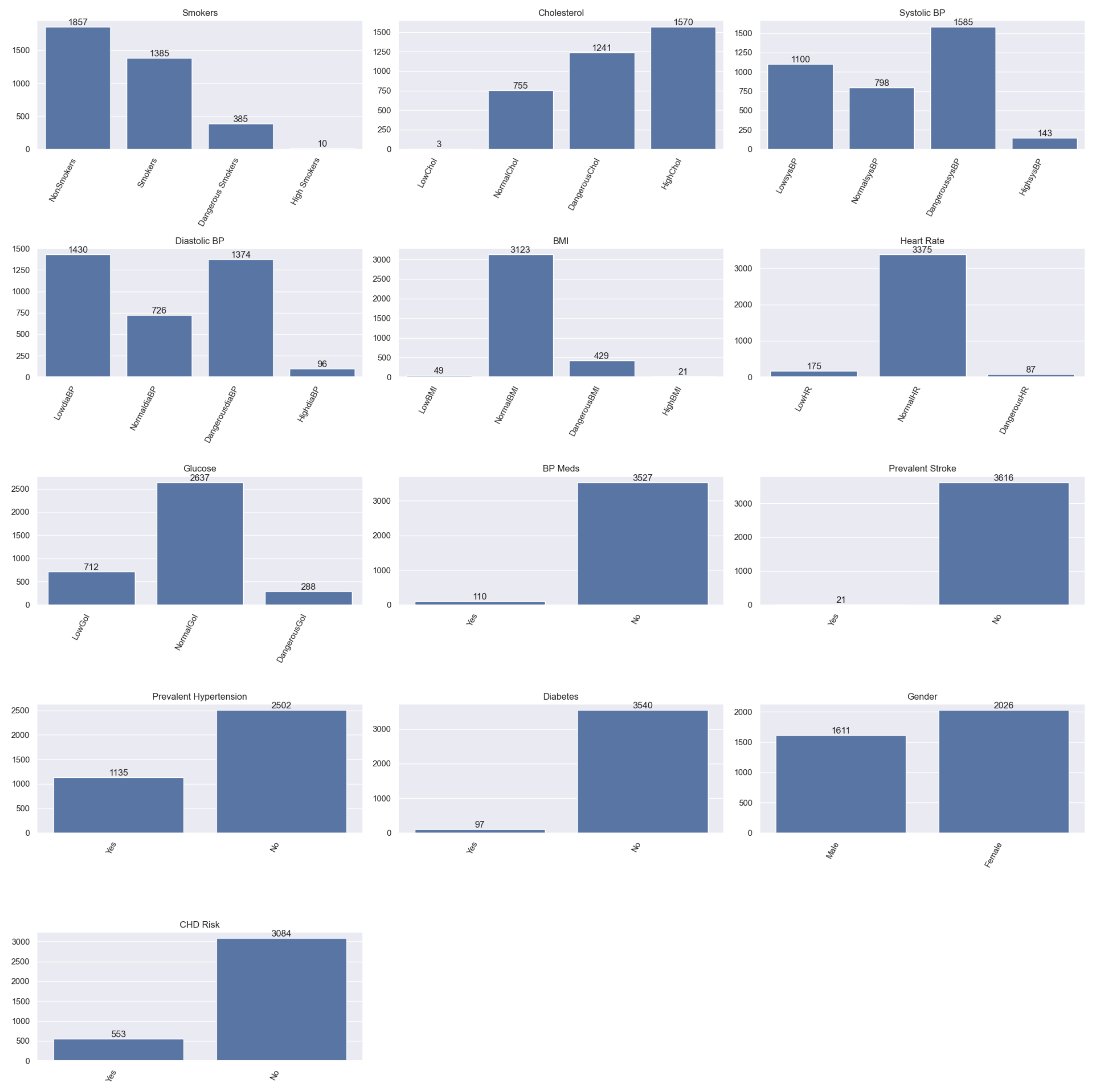
- (barplot)

[399]:
```
from matplotlib.ticker import FixedLocator

# Create dictionaries for each category
Smokers = {'NonSmokers': df[(df.cigsPerDay == 0)]['cigsPerDay'].count(),
           'Smokers': df[(df.cigsPerDay > 0) & (df.cigsPerDay <=␣
 ↪20)]['cigsPerDay'].count(),
           'Dangerous Smokers': df[(df.cigsPerDay > 20) & (df.cigsPerDay <=␣
 ↪50)]['cigsPerDay'].count(),
           'High Smokers': df[(df.cigsPerDay > 50)]['cigsPerDay'].count()}

Chol = {'LowChol': df[(df.totChol < 125)]['totChol'].count(),
        'NormalChol': df[(df.totChol > 125) & (df.totChol <= 200)]['totChol'].
 ↪count(),
        'DangerousChol': df[(df.totChol > 200) & (df.totChol <=␣
↪239)]['totChol'].count(),
        'HighChol': df[(df.totChol > 240)]['totChol'].count()}

SysBP = {'LowsysBP': df[(df.sysBP < 120)]['sysBP'].count(),
         'NormalsysBP': df[(df.sysBP >= 120) & (df.sysBP <= 129)]['sysBP'].
↪count(),
         'DangeroussysBP': df[(df.sysBP >= 130) & (df.sysBP <= 179)]['sysBP'].
↪count(),
         'HighsysBP': df[(df.sysBP >= 180)]['sysBP'].count()}

DiaBP = {'LowdiaBP': df[(df.diaBP < 80)]['diaBP'].count(),
         'NormaldiaBP': df[(df.diaBP >= 80) & (df.diaBP <= 84)]['diaBP'].
↪count(),
         'DangerousdiaBP': df[(df.diaBP >= 85) & (df.diaBP <= 109)]['diaBP'].
↪count(),
         'HighdiaBP': df[(df.diaBP >= 110)]['diaBP'].count()}

BMI = {'LowBMI': df[(df.BMI < 18.5)]['BMI'].count(),
       'NormalBMI': df[(df.BMI >= 18.5) & (df.BMI <= 29.9)]['BMI'].count(),
       'DangerousBMI': df[(df.BMI >= 30) & (df.BMI <= 39.9)]['BMI'].count(),
       'HighBMI': df[(df.BMI >= 40)]['BMI'].count()}

HeartRate = {'LowHR': df[(df.heartRate < 60)]['heartRate'].count(),
             'NormalHR': df[(df.heartRate >= 60) & (df.heartRate <=␣
↪100)]['heartRate'].count(),
             'DangerousHR': df[(df.heartRate > 100)]['heartRate'].count()}


Glo = {'LowGol': df[(df.glucose < 70)]['glucose'].count(),
       'NormalGol': df[(df.glucose >= 70) & (df.glucose <= 100)]['glucose'].
↪count(),
       'DangerousGol': df[(df.glucose > 100)]['glucose'].count()}

BPMeds = {'Yes': df[(df.BPMeds == 'yes')]['BPMeds'].count(),
          'No': df[(df.BPMeds == 'no')]['BPMeds'].count()}

prevalentStroke = {'Yes': df[(df.prevalentStroke == 'yes')]['prevalentStroke'].
↪count(),
                   'No': df[(df.prevalentStroke == 'no')]['prevalentStroke'].
↪count()}

prevalentHyp = {'Yes': df[(df.prevalentHyp == 'yes')]['prevalentHyp'].count(),
                'No': df[(df.prevalentHyp == 'no')]['prevalentHyp'].count()}

gender = {'Male': df[df['sex'] == 'male']['sex'].count(),
          'Female': df[df['sex'] == 'female']['sex'].count()}

CHDRisk = {'Yes': df[(df.CHDRisk == 'yes')]['CHDRisk'].count(),
           'No': df[(df.CHDRisk == 'no')]['CHDRisk'].count()}

Diabetes = {'Yes': df[(df.diabetes == 'yes')]['diabetes'].count(),
            'No': df[(df.diabetes == 'no')]['diabetes'].count()}

# List of dictionaries and titles
dicts = [Smokers, Chol, SysBP, DiaBP, BMI, HeartRate, Glo, BPMeds,␣
 ↪prevalentStroke, prevalentHyp, Diabetes, gender, CHDRisk]
dicts_title = ['Smokers', 'Cholesterol', 'Systolic BP', 'Diastolic BP', 'BMI',␣
 ↪'Heart Rate', 'Glucose', 'BP Meds', 'Prevalent Stroke', 'Prevalent␣
 ↪Hypertension', 'Diabetes', 'Gender', 'CHD Risk']

# Plotting
plt.figure(figsize=(20, 20))

for e, dict_data in enumerate(dicts):
    d = pd.DataFrame(dict_data.items(), columns=['Classification', 'Value'])


    print(f" {dicts_title[e]}")
    print(d)
    print('=' * 50)


    plt.subplot(5, 3, e+1)
    ax = sns.barplot(data=d, x='Classification', y='Value')
    ax.bar_label(ax.containers[0])# to show count outside plot
    ax.set_title(dicts_title[e])
    ax.set_xlabel('')
    ax.set_ylabel('')


    ax.set_xticks(range(len(d['Classification'])))
    ax.set_xticklabels(d['Classification'], rotation=62, ha='right')

plt.tight_layout()
plt.show()
```

```
 Smokers
      Classification  Value
0         NonSmokers   1857
1            Smokers   1385
2  Dangerous Smokers    385
3       High Smokers     10
```

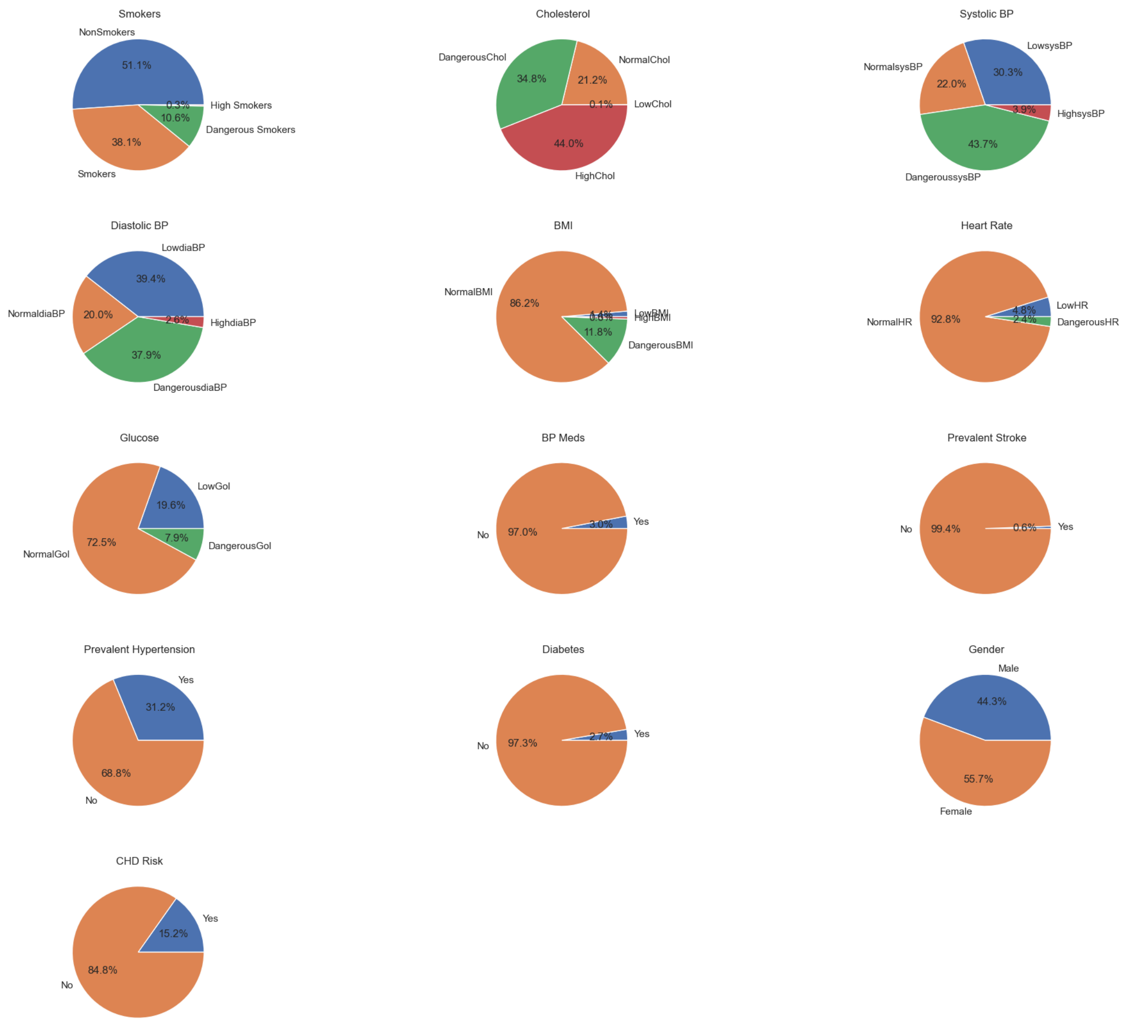
```
==================================================
 Cholesterol
  Classification  Value
0        LowChol      3
1     NormalChol    755
2  DangerousChol   1241
3       HighChol   1570
==================================================
 Systolic BP
   Classification  Value
0        LowsysBP   1100
1     NormalsysBP    798
2  DangeroussysBP   1585
3       HighsysBP    143
==================================================
 Diastolic BP
   Classification  Value
0        LowdiaBP   1430
1     NormaldiaBP    726
2  DangerousdiaBP   1374
3       HighdiaBP     96
==================================================
 BMI
  Classification  Value
0         LowBMI     49
1      NormalBMI   3123
2   DangerousBMI    429
3        HighBMI     21
==================================================
 Heart Rate
  Classification  Value
0          LowHR    175
1       NormalHR   3375
2    DangerousHR     87
==================================================
 Glucose
  Classification  Value
0         LowGol    712
1      NormalGol   2637
2   DangerousGol    288
==================================================
 BP Meds
  Classification  Value
0            Yes    110
1             No   3527
==================================================
 Prevalent Stroke
  Classification  Value
```

```
0            Yes     21
1             No   3616
==================================================
 Prevalent Hypertension
  Classification  Value
0            Yes   1135
1             No   2502
==================================================
 Diabetes
  Classification  Value
0            Yes     97
1             No   3540
==================================================
 Gender
  Classification  Value
0           Male   1611
1         Female   2026
==================================================
 CHD Risk
  Classification  Value
0            Yes    553
1             No   3084
==================================================
```



- (pie)

```
[400]: plt.figure(figsize=(20,20))
       for i , dict_col in enumerate(dicts):
           d=pd.DataFrame(dict_col.items(),columns=['Classification','Value'])
           plt.subplot(6,3,i+1)
           plt.pie(d['Value'],labels=d['Classification'],autopct='%1.1f%%')
           plt.tight_layout()
           plt.title(f'\n \n {dicts_title[i]}')
       plt.show()
```

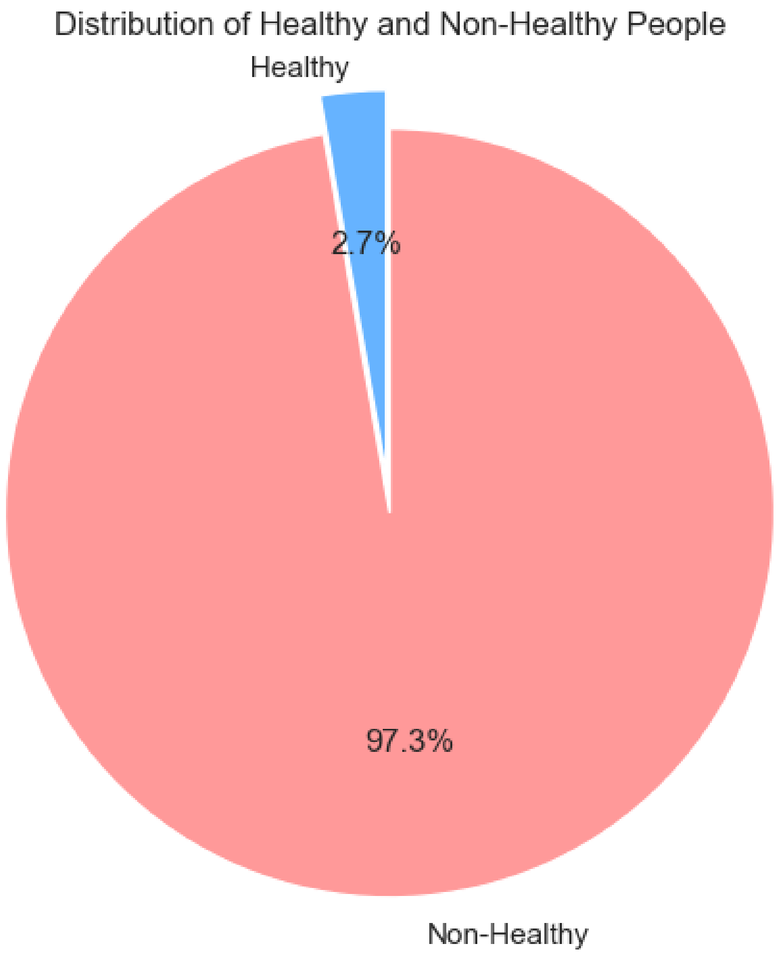


[401]:
```
non_healthy_people=len(df)-length_healthy_people
non_healthy_people
```

[401]:
```
3537
```

Distribution of Healthy vs. Non-Healthy Individuals

[402]:
```
size=[length_healthy_people,non_healthy_people]
labels=[' \n    Healthy','Non-Healthy']
colors = ['#66b3ff', '#ff9999']

plt.figure(figsize=(6,6))
plt.pie(size,labels=labels,colors=colors,autopct='%1.1f%%',startangle=90,␣
  ↪explode=(0, 0.1))
plt.title('Distribution of Healthy and Non-Healthy People')
plt.axis('equal')
plt.show()
```



[403]:
```
df.columns
```

[403]:
```
Index(['sex', 'education', 'smokingStatus', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'CHDRisk', 'age_group'],
      dtype='object')
```

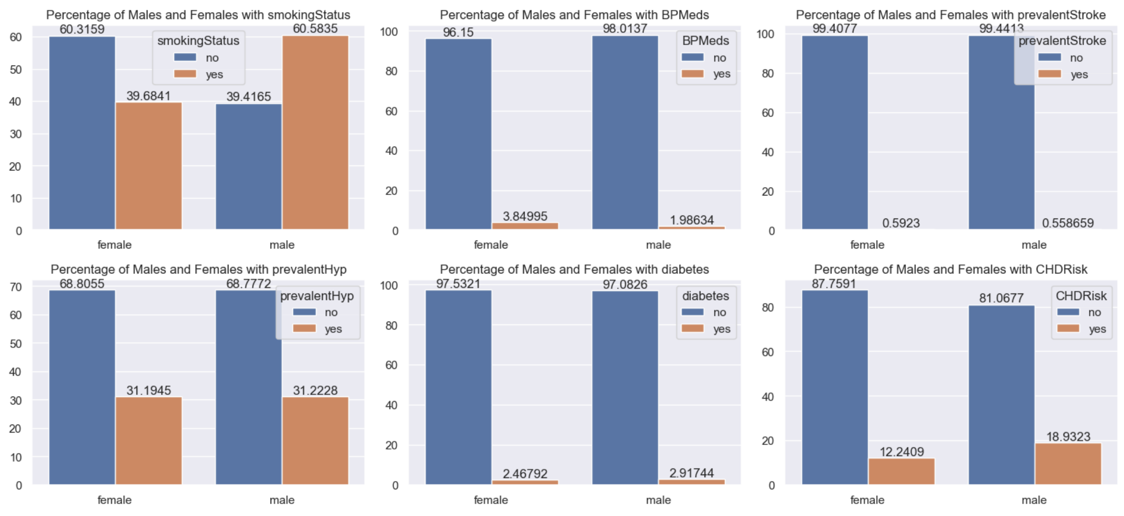
percentage of males and females with various health conditions

[405]:
```
disease_columns =['smokingStatus', 'BPMeds', 'prevalentStroke', 'prevalentHyp',␣
 ↪'diabetes', 'CHDRisk']

plt.figure(figsize=(15,10))
```

```
for i ,col in enumerate(disease_columns):
    gender_disease_count=df.groupby(['sex',col]).size().
 ↪reset_index(name='count')
#     print(gender_disease_count)
    gender_total=df.groupby('sex').size().reset_index(name='total')
#     print(gender_total)
    merged_df=pd.merge(gender_disease_count,gender_total,on='sex')
#     print(merged_df)
    merged_df['Percentage']=(merged_df['count'] / merged_df['total'])*100
#     print(merged_df)
    print(f"\nPercentage of Males and Females with {col}:")


    #iterrows() function returns both the index and the row data.
    #If you are only interested in the row and do not need the index, you can␣
 ↪use _ to indicate that you are intentionally ignoring the index.
    for _, row in merged_df.iterrows():
        print(f"Gender: {row['sex']}, Disease Status: {row[col]}, Percentage:␣
 ↪{row['Percentage']:.2f}%")

    plt.subplot(3, 3, i+1)
    ax=sns.barplot(data=merged_df, x='sex', y='Percentage', hue=col)
    ax.bar_label(ax.containers[0])# to show count outside plot
    ax.bar_label(ax.containers[1])

    ax.set_title(f'Percentage of Males and Females with {col}')
    ax.set_ylabel('')
    ax.set_xlabel('')

plt.tight_layout()
plt.show()
```

```
Percentage of Males and Females with smokingStatus:
Gender: female, Disease Status: no, Percentage: 60.32%
Gender: female, Disease Status: yes, Percentage: 39.68%
Gender: male, Disease Status: no, Percentage: 39.42%
Gender: male, Disease Status: yes, Percentage: 60.58%

Percentage of Males and Females with BPMeds:
Gender: female, Disease Status: no, Percentage: 96.15%
Gender: female, Disease Status: yes, Percentage: 3.85%
Gender: male, Disease Status: no, Percentage: 98.01%
Gender: male, Disease Status: yes, Percentage: 1.99%
```

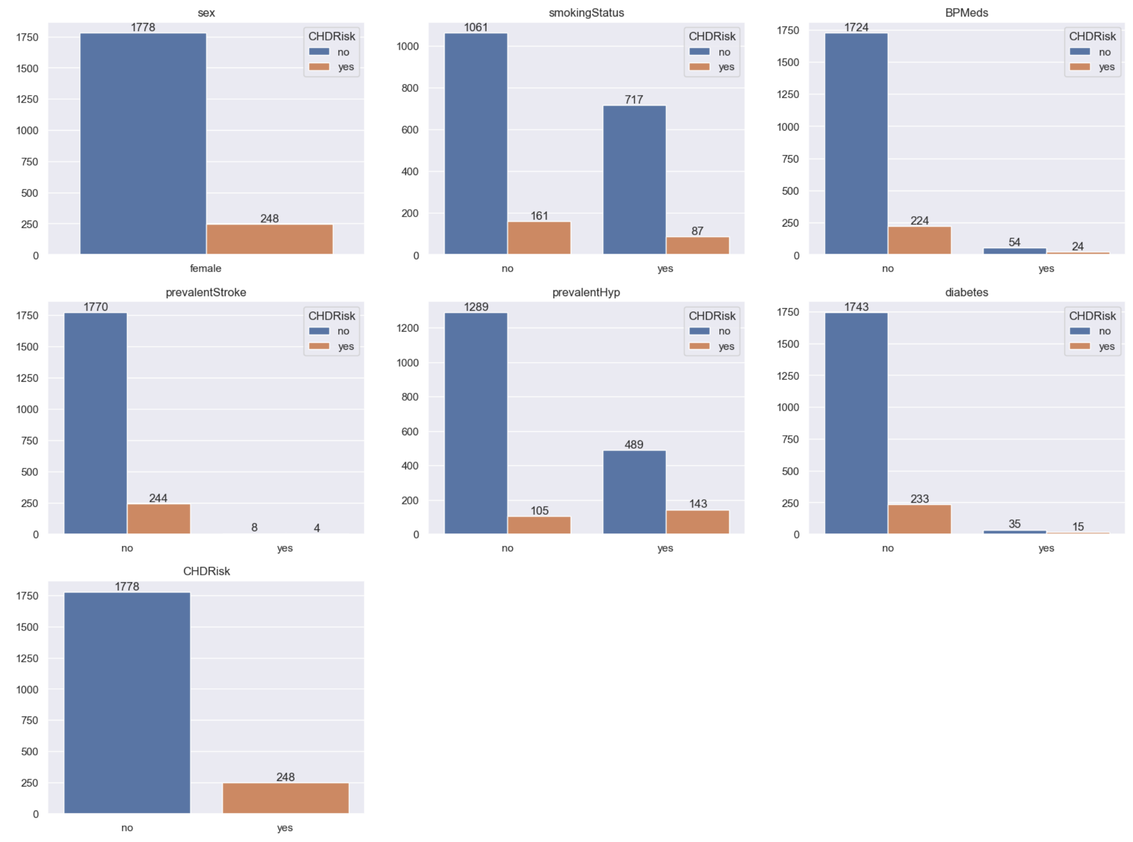
```
Percentage of Males and Females with prevalentStroke:
Gender: female, Disease Status: no, Percentage: 99.41%
Gender: female, Disease Status: yes, Percentage: 0.59%
Gender: male, Disease Status: no, Percentage: 99.44%
Gender: male, Disease Status: yes, Percentage: 0.56%

Percentage of Males and Females with prevalentHyp:
Gender: female, Disease Status: no, Percentage: 68.81%
Gender: female, Disease Status: yes, Percentage: 31.19%
Gender: male, Disease Status: no, Percentage: 68.78%
Gender: male, Disease Status: yes, Percentage: 31.22%

Percentage of Males and Females with diabetes:
Gender: female, Disease Status: no, Percentage: 97.53%
Gender: female, Disease Status: yes, Percentage: 2.47%
Gender: male, Disease Status: no, Percentage: 97.08%
Gender: male, Disease Status: yes, Percentage: 2.92%

Percentage of Males and Females with CHDRisk:
Gender: female, Disease Status: no, Percentage: 87.76%
Gender: female, Disease Status: yes, Percentage: 12.24%
Gender: male, Disease Status: no, Percentage: 81.07%
Gender: male, Disease Status: yes, Percentage: 18.93%
```



relationship between various health conditions and CHDRisk among females

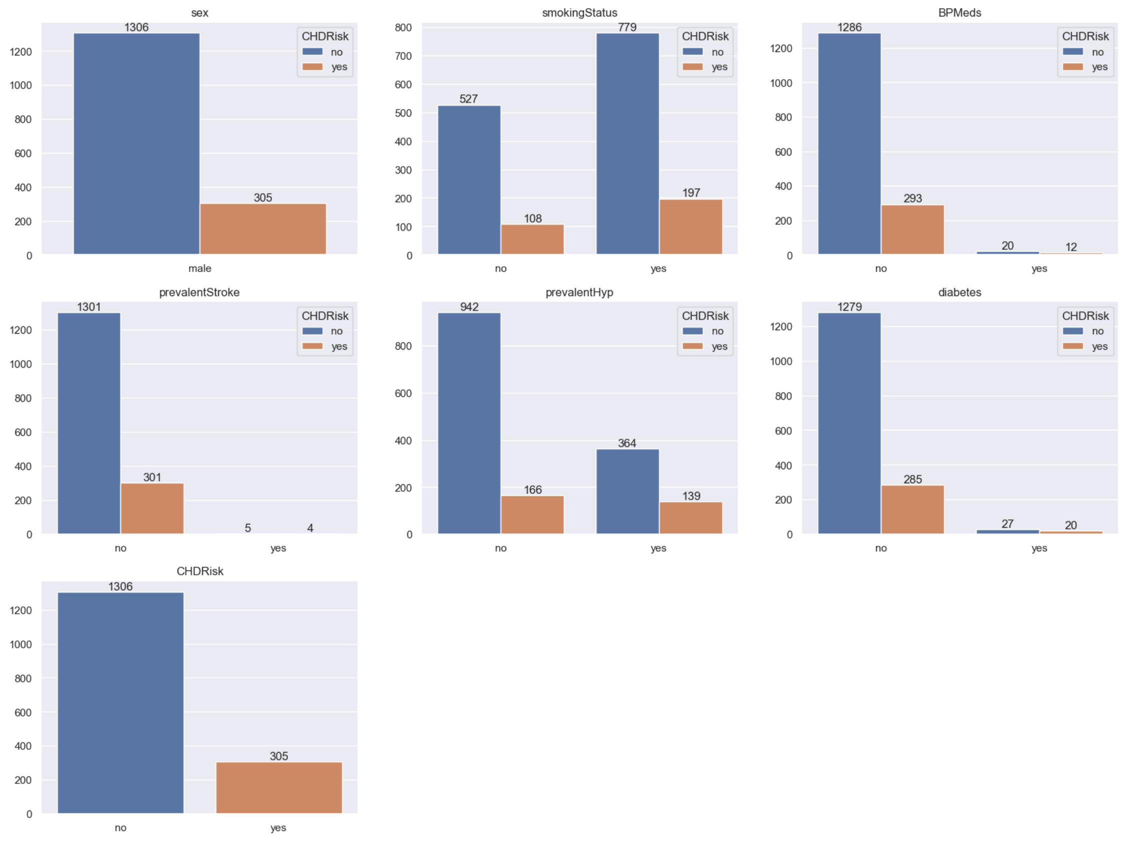
[406]:
```
disease = df[['sex','smokingStatus', 'BPMeds', 'prevalentStroke',␣
↪'prevalentHyp', 'diabetes', 'CHDRisk']]

gender_female=disease[disease.sex == 'female']
```

```
plt.figure(figsize=(20,20))
for i ,col in enumerate(gender_female):
    plt.subplot(4,3,i+1)
    ax=sns.countplot(data=gender_female,x=col ,hue='CHDRisk')
    ax.bar_label(ax.containers[0])
    ax.bar_label(ax.containers[1])
    ax.set_title('\n\n' + col)
    ax.set_xlabel('')
    ax.set_ylabel('')
```



relationship between various health conditions and CHDRisk among males

```
[407]: disease = df[['sex','smokingStatus', 'BPMeds', 'prevalentStroke',␣
        ↪'prevalentHyp', 'diabetes', 'CHDRisk']]

gender_male=disease[disease.sex == 'male']

plt.figure(figsize=(20,20))
for i ,col in enumerate(gender_male):
    plt.subplot(4,3,i+1)
    ax=sns.countplot(data=gender_male,x=col ,hue='CHDRisk')
    ax.bar_label(ax.containers[0])
    ax.bar_label(ax.containers[1])
    ax.set_title('\n\n' + col)
    ax.set_xlabel('')
    ax.set_ylabel('')
```



11 statistical tests

1-Chi-Square Test for Categorical Variables

- Chi-Square Test Use this to test for independence between two categorical variables.show if sex and CHDRisk are associated.

[408]:
```
# Chi-square test
from scipy.stats import chi2_contingency
contingency = pd.crosstab(df['sex'], df['CHDRisk'])
chi2, p, dof, expected = chi2_contingency(contingency)
print(f"Chi-square: {chi2}, p-value: {p}")
```

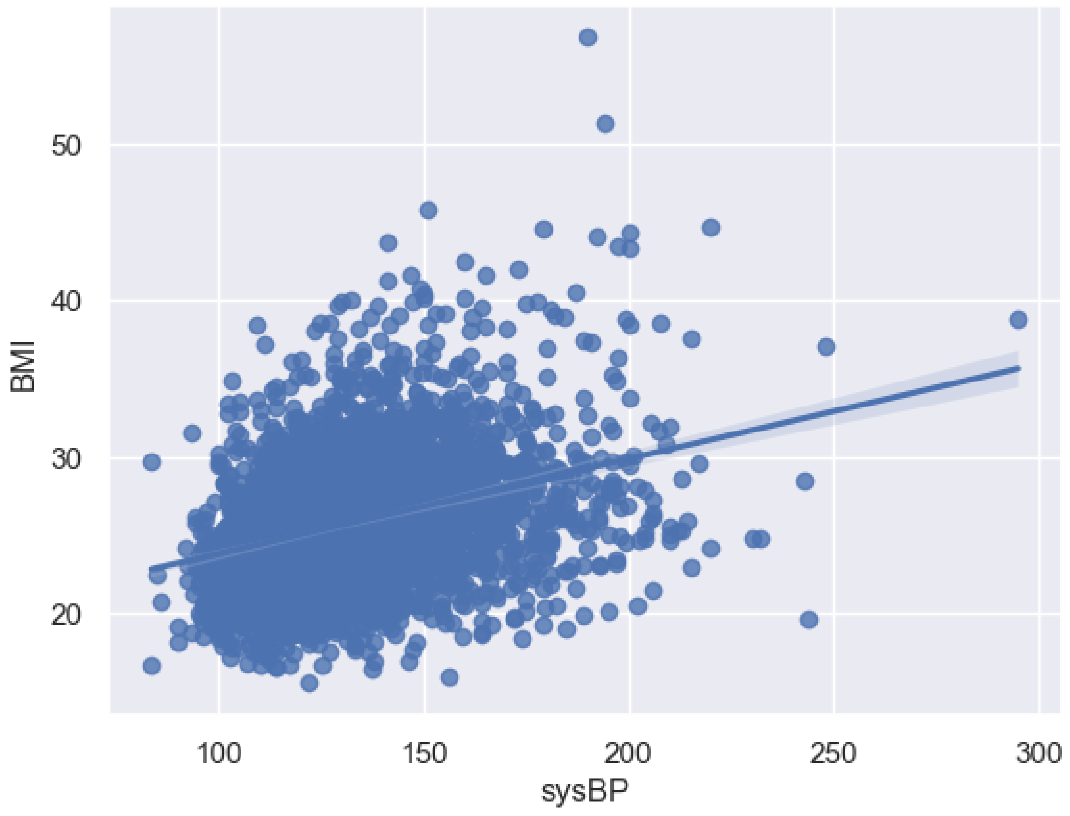
```
if p < 0.05:
    print("There is a significant relationship between age_group and CHDRisk")
else:
    print("No significant relationship between age_group and CHDRisk")
```

```
Chi-square: 30.64922901316927, p-value: 3.091494190770845e-08
There is a significant relationship between age_group and CHDRisk
```

2-Pearson Correlation Coefficient: For testing the linear relationship between two continuous variables

[409]:
```
from scipy.stats import pearsonr
corr ,p_val =pearsonr(df['sysBP'],df['BMI'])
print(f'Pearson Correlation={corr}, p={p}')
if p < 0.05:
    print('There is a significant correlation between sysBP and BMI')
else:
    print('No significant correlation between sysBP and BMI')

sns.regplot(data=df,x='sysBP', y='BMI')
plt.show()
```

```
Pearson Correlation=0.3292817576370576, p=3.091494190770845e-08
There is a significant correlation between sysBP and BMI
```



12 Logistic Regression Significance Test:

For testing whether predictor variables in a logistic regression are significant

[410]:
```
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()
df['smokingStatus']=le.fit_transform(df['smokingStatus'])
df['BPMeds']=le.fit_transform(df['BPMeds'])
df['prevalentStroke']=le.fit_transform(df['prevalentStroke'])
df['prevalentHyp']=le.fit_transform(df['prevalentHyp'])
df['diabetes']=le.fit_transform(df['diabetes'])
df['age_group'] = le.fit_transform(df['age_group'])
```

[413]:
```
df['sex']=df['sex'].replace('female',0)
df['sex']=df['sex'].replace('male',1)

df['CHDRisk']=df['CHDRisk'].replace('no',0)
df['CHDRisk']=df['CHDRisk'].replace('yes',1)

df.sample(5)
```

[413]:
```
      sex  education  smokingStatus  cigsPerDay  BPMeds  prevalentStroke  \
2585    1          1              0           0       0                0
425     1          3              1          20       0                0
830     0          4              0           0       0                0
2413    0          2              0           0       0                0
1603    0          1              0           0       1                0

      prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  heartRate  glucose  \
2585             0         0      188 105.00  65.00  22.85         63       76
425              0         0      340 134.00  89.50  21.91         50       72
830              1         0      268 151.00  98.00  20.34         72       60
2413             0         0      215 110.00  70.00  19.64         70       87
1603             1         1      294 195.00  90.00  27.73         72      127

      CHDRisk  age_group
2585        0          0
425         1          1
830         0          1
2413        0          0
1603        0          1
```

```
[414]: import statsmodels.api as sm

       X=df.drop(columns=['CHDRisk'])
       X=sm.add_constant(X)
       y=df['CHDRisk']
```

```
[415]: logit_model=sm.Logit(y,X)
```

```
[416]: results=logit_model.fit()
       print(results.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.381751
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                CHDRisk   No. Observations:                 3637
Model:                          Logit   Df Residuals:                     3621
Method:                           MLE   Df Model:                           15
Date:                Mon, 09 Sep 2024   Pseudo R-squ.:                  0.1044
Time:                        13:50:17   Log-Likelihood:                -1388.4
converged:                       True   LL-Null:                       -1550.3
Covariance Type:            nonrobust   LLR p-value:                 6.669e-60
================================================================================
===
                      coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---
const              -6.0047      0.636     -9.435      0.000      -7.252
-4.757
sex                 0.5872      0.108      5.416      0.000       0.375
0.800
education          -0.0796      0.050     -1.607      0.108      -0.177
0.017
smokingStatus       0.0172      0.157      0.109      0.913      -0.291
0.325
cigsPerDay          0.0161      0.006      2.584      0.010       0.004
0.028
BPMeds              0.1488      0.237      0.628      0.530      -0.316
0.613
prevalentStroke     0.8417      0.486      1.732      0.083      -0.111
1.794
prevalentHyp        0.2511      0.138      1.817      0.069      -0.020
0.522
diabetes            0.1092      0.317      0.345      0.730      -0.512
0.730
totChol             0.0030      0.001      2.720      0.007       0.001
```

```
0.005
sysBP              0.0201      0.004      5.346      0.000       0.013
0.028
diaBP             -0.0092      0.006     -1.434      0.152      -0.022
0.003
BMI                0.0079      0.013      0.618      0.537      -0.017
0.033
heartRate         -0.0048      0.004     -1.141      0.254      -0.013
0.003
glucose            0.0073      0.002      3.243      0.001       0.003
0.012
age_group          0.6689      0.098      6.823      0.000       0.477
0.861
================================================================================
===
```

if the p-value of a feature is less than 0.05, it means the feature is significant. . remove all features that p-value >0.05

[417]:
```
X.columns
```

[417]:
```
Index(['const', 'sex', 'education', 'smokingStatus', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'age_group'],
      dtype='object')
```

[418]:
```
X=df[['sex','cigsPerDay','totChol','glucose','sysBP','age_group']]
X=sm.add_constant(X)
y=df['CHDRisk']
logit_model=sm.Logit(y,X)
results=logit_model.fit()
print(results.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.383639
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                CHDRisk   No. Observations:                 3637
Model:                          Logit   Df Residuals:                     3630
Method:                           MLE   Df Model:                            6
Date:                Mon, 09 Sep 2024   Pseudo R-squ.:                 0.09996
Time:                        13:50:53   Log-Likelihood:                -1395.3
converged:                       True   LL-Null:                       -1550.3
Covariance Type:            nonrobust   LLR p-value:                 6.124e-64
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -7.1055      0.408    -17.414      0.000      -7.905      -6.306
```

```
sex            0.5901      0.106      5.567      0.000       0.382       0.798
cigsPerDay     0.0154      0.004      3.756      0.000       0.007       0.023
totChol        0.0030      0.001      2.701      0.007       0.001       0.005
glucose        0.0079      0.002      4.602      0.000       0.005       0.011
sysBP          0.0205      0.002      9.669      0.000       0.016       0.025
age_group      0.7189      0.096      7.510      0.000       0.531       0.906
==============================================================================
```

[419]:
```
results.params.index
```

[419]:
```
Index(['const', 'sex', 'cigsPerDay', 'totChol', 'glucose', 'sysBP',
       'age_group'],
      dtype='object')
```

[420]:
```
results.params.index[2:]
```

[420]:
```
Index(['cigsPerDay', 'totChol', 'glucose', 'sysBP', 'age_group'],
dtype='object')
```

[423]:
```
for param in results.params.index[2:]:
    odds_ratio=np.exp(results.params[param])#  np.exp Converts the coefficient␣
 ↪into the odds ratio, making the result easier to interpret.
    print(f"For each 1 unit increase in {param}, the odds of CHDRisk increase␣
 ↪by {round(odds_ratio, 2)} times, holding all else constant")
```

```
For each 1 unit increase in cigsPerDay, the odds of CHDRisk increase by 1.02
times, holding all else constant
For each 1 unit increase in totChol, the odds of CHDRisk increase by 1.0 times,
holding all else constant
For each 1 unit increase in glucose, the odds of CHDRisk increase by 1.01 times,
holding all else constant
For each 1 unit increase in sysBP, the odds of CHDRisk increase by 1.02 times,
holding all else constant
For each 1 unit increase in age_group, the odds of CHDRisk increase by 2.05
times, holding all else constant
```

13 Prediction

[435]:
```
X=df[['sex','cigsPerDay','totChol','glucose','sysBP','age_group']]
y=df['CHDRisk']
```

[436]:
```
from  sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
 ↪2,random_state=42,stratify=y)

scaler=StandardScaler()
```

```
x_train_scaled=scaler.fit_transform(X_train)
x_test_scaled=scaler.transform(X_test)
```

[456]:
```
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier


models={
    'logistic_Model':LogisticRegression(),
    'random_model':RandomForestClassifier(),
    'gradient_model':GradientBoostingClassifier(),
    'knn':KNeighborsClassifier(),
        'svm_model':SVC()

}
```

14 GridSearch : to find the best hyperparameters

[461]:
```
from sklearn.model_selection import GridSearchCV

# Define the parameter grids for each model
param_grids = {
    "logistic_Model": {
        'C': [0.01, 0.1, 1, 10, 100],
        'penalty': ['l1', 'l2'],
        'solver': ['liblinear']
    },


    "random_model": {
        'n_estimators': [50, 100, 200],
        'max_depth': [None, 10, 20, 30],
        'min_samples_split': [2, 5, 10]
    },

    "gradient_model": {
        'n_estimators': [50, 100, 200],
        'learning_rate': [0.01, 0.1, 0.2],
        'max_depth': [3, 5, 7]
    },
    "knn": {
        'n_neighbors': [3, 5, 7, 9],
        'weights': ['uniform', 'distance']
    },
     "svm_model": {
        'C': [0.1, 1, 10, 100],
        'kernel': ['linear', 'rbf'],
        'gamma': ['scale', 'auto']
    }
}

# Function to perform GridSearchCV
def perform_grid_search(model, param_grid):
    grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy',␣
 ↪n_jobs=-1)
    grid_search.fit(x_train_scaled, y_train)
    return grid_search

#  GridSearchCV for each model
for name, model in models.items():
    print(f"Performing Grid Search for {name}...")
    grid_search = perform_grid_search(model, param_grids[name])
    print(f"Best parameters for {name}: {grid_search.best_params_}")
    print(f"Best accuracy for {name}: {grid_search.best_score_}")
    print("-" * 50)
```

```
Performing Grid Search for logistic_Model…
Best parameters for logistic_Model: {'C': 0.01, 'penalty': 'l2', 'solver':
'liblinear'}
Best accuracy for logistic_Model: 0.8514937511459681
--------------------------------------------------
Performing Grid Search for random_model…
Best parameters for random_model: {'max_depth': 20, 'min_samples_split': 10,
'n_estimators': 200}
Best accuracy for random_model: 0.8483962359008939
--------------------------------------------------
Performing Grid Search for gradient_model…
Best parameters for gradient_model: {'learning_rate': 0.01, 'max_depth': 5,
'n_estimators': 50}
Best accuracy for gradient_model: 0.8484015591083036
--------------------------------------------------
Performing Grid Search for knn…
Best parameters for knn: {'n_neighbors': 9, 'weights': 'uniform'}
Best accuracy for knn: 0.8470258057265883
--------------------------------------------------
Performing Grid Search for svm_model…
Best parameters for svm_model: {'C': 1, 'gamma': 'auto', 'kernel': 'rbf'}
Best accuracy for svm_model: 0.8490888443316713
--------------------------------------------------
```

15 Data is Imbalanced:

[457]:
```python
from sklearn.metrics import confusion_matrix
model=SVC()
model.fit(x_train_scaled,y_train)
y_pred=model.predict(x_test_scaled)
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
[[617   0]
 [108   3]]
```

Total Negative Cases (TN + FP): 617 + 0 = 617 Total Positive Cases (FN + TP): 108 + 3 = 111 .

The imbalance is evident because the number of samples in the majority class (no CHD risk) is much larger than the number of samples in the minority class (CHD risk). Specifically, there are 617 instances of the majority class compared to only 111 instances of the minority class .

So ,we will use classification_report not Confision Matrix

[458]:
```python
print(df['CHDRisk'].value_counts())
```

```
CHDRisk
0    3084
1     553
Name: count, dtype: int64
```

it is also here number of 0 class > number of 1 class

[459]:
```python
from sklearn.metrics import accuracy_score, classification_report

for name , model in models.items():
    model.fit(x_train_scaled,y_train)
    y_pred=model.predict(x_test_scaled)
    print(f"{name}:")
    print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
    print(classification_report(y_test, y_pred))
    print("-" * 50)
```

```
logistic_Model:
Accuracy: 0.853021978021978
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       617
           1       0.64      0.08      0.14       111

    accuracy                           0.85       728
   macro avg       0.75      0.54      0.53       728
weighted avg       0.82      0.85      0.80       728
```

```
--------------------------------------------------
random_model:
Accuracy: 0.8475274725274725
              precision    recall  f1-score   support

           0       0.86      0.97      0.92       617
           1       0.50      0.15      0.23       111

    accuracy                           0.85       728
   macro avg       0.68      0.56      0.57       728
weighted avg       0.81      0.85      0.81       728

--------------------------------------------------
gradient_model:
Accuracy: 0.8502747252747253
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       617
           1       0.56      0.08      0.14       111

    accuracy                           0.85       728
   macro avg       0.71      0.53      0.53       728
weighted avg       0.81      0.85      0.80       728

--------------------------------------------------
knn:
Accuracy: 0.8543956043956044
              precision    recall  f1-score   support

           0       0.87      0.98      0.92       617
           1       0.57      0.18      0.27       111

    accuracy                           0.85       728
   macro avg       0.72      0.58      0.60       728
weighted avg       0.82      0.85      0.82       728

--------------------------------------------------
svm_model:
Accuracy: 0.8516483516483516
              precision    recall  f1-score   support

           0       0.85      1.00      0.92       617
           1       1.00      0.03      0.05       111

    accuracy                           0.85       728
   macro avg       0.93      0.51      0.49       728
weighted avg       0.87      0.85      0.79       728
```

----

16 Summary of the Logistic Model Results:

- **Precision** tells you how accurate your positive predictions are.
- **Recall** tells you how well your model detects all actual positives.
- **F1-Score** provides a balanced measure of precision and recall, especially useful when dealing with imbalanced classes.

. - **Accuracy**: The model has an overall accuracy of **85.3%**, meaning it correctly predicts 85.3% of the instances.

- **Class `0` (majority class)**:
  - **Precision**: **86%** of the instances predicted as class `0` are correct.
  - **Recall**: **99%** of the actual class `0` instances are correctly identified.
  - **F1-Score**: **92%**, indicating excellent performance for class `0`.
- **Class `1` (minority class)**:
  - **Precision**: **64%** of the instances predicted as class `1` are correct.
  - **Recall**: Only **8%** of the actual class `1` instances are identified correctly.
  - **F1-Score**: **14%**, showing poor performance in detecting class `1` due to the low recall.

. - **Macro Avg** (unweighted average): - **Precision**: **75%**, averaging the precision of both classes equally. - **Recall**: **54%**, reflecting the model's lower performance in detecting class `1`. - **F1-Score**: **53%**, showing the balance between precision and recall.

- **Weighted Avg** (weighted by class frequency):
  - **Precision**: **82%**, with more weight given to class `0`.
  - **Recall**: **85%**, influenced by the high recall of class `0`.
  - **F1-Score**: **80%**, emphasizing good overall performance but issues with class `1`.

16.0.1 Key Issue:

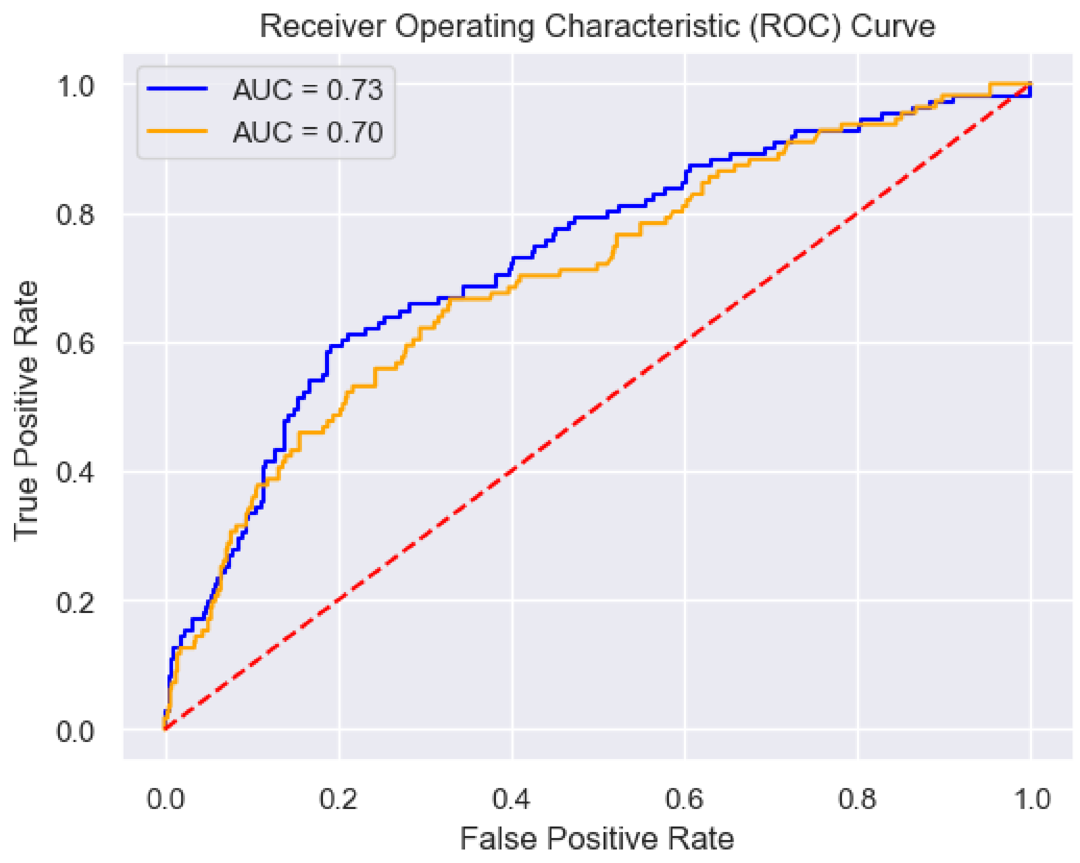
In your results, the model performs very well for class `0` but poorly for class `1`, showing that precision and recall are not balanced for the minority class.

17 Compare the ROC Curves of two Models.

[460]:
```
from sklearn.metrics import roc_auc_score, roc_curve

model_1=LogisticRegression()
model_2=GradientBoostingClassifier()

model_1.fit(x_train_scaled,y_train)
y_pred_model_1 = model_1.predict(x_test_scaled)
y_prob_model_1 = model_1.predict_proba(x_test_scaled)[:, 1]  # Probabilities␣
 ↪for AUC


model_2.fit(x_train_scaled,y_train)
```

```
y_pred_model_2 = model_2.predict(x_test_scaled)
y_prob_model_2 = model_2.predict_proba(x_test_scaled)[:, 1]  # Probabilities␣
 ↪for AUC

# Calculate ROC AUC score
roc_auc_1 = roc_auc_score(y_test, y_prob_model_1)
print(f'AUC: {roc_auc_1}')

roc_auc_2 = roc_auc_score(y_test, y_prob_model_2)
print(f'AUC: {roc_auc_2}')

# Plot ROC curve
fpr1, tpr1, thresholds1 = roc_curve(y_test, y_prob_model_1)
fpr2, tpr2, thresholds2 = roc_curve(y_test, y_prob_model_2)

plt.plot(fpr1, tpr1, color='blue', label=f'AUC = {roc_auc_1:.2f}')
plt.plot(fpr2, tpr2, color='orange', label=f'AUC = {roc_auc_2:.2f}')

plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```

```
AUC: 0.7311752595383065
AUC: 0.7025274869683298
```



LogisticRegression is better than gradient

What is an ROC Curve?

The ROC (Receiver Operating Characteristic) Curve is a graphical representation of the performance of a classification model at different threshold values. It plots two metrics:

True Positive Rate (Recall) on the y-axis:

This measures how many actual positives (True Positives) are correctly identified.

`<p><i>True Positive Rate (TPR) = True Positives / (True Positives + False Negatives)</i></p>`

False Positive Rate on the x-axis:

This measures how many actual negatives are incorrectly classified as positives.

`<p><i>False Positive Rate (FPR) = False Positives / (False Positives + True Negatives)</i></p>`

Key Points:

The closer the ROC curve is to the top-left corner, the better the model's performance.

A random classifier would produce a diagonal line (FPR = TPR), representing an AUC of 0.5.

In summary, the ROC curve helps visualize the trade-off between the True Positive Rate and the False Positive Rate, while the AUC quantifies the overall performance of the model. A higher AUC means a better performing model.

Interpretation of AUC (Area Under the Curve):

AUC $= 1.0$: The model is perfect, distinguishing between classes with 100% accuracy.

0.9 AUC < 1.0: The model has excellent performance, meaning it is highly accurate in distinguishing between the classes.

0.8 AUC < 0.9: The model has good performance, but not perfect.

0.7 AUC < 0.8: The model has fair performance. It can distinguish between the classes better than random guessing, but there's room for improvement.

0.5 AUC < 0.7: The model has poor performance, only slightly better than random guessing.

AUC $= 0.5$: The model has no discriminatory power and is equivalent to random guessing.

```
[ ]:
```