

To perform and find the accuracy of Naive bayes Classifier

```
In [ ]: #Name: Shruti .P. Arsode
        #Roll no.: 03
        #Sec:'A'
```

```
In [ ]: import pandas as pd
import os
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

```
In [2]: os.getcwd()
```

```
Out[2]: 'C:\\Users\\lenovo'
```

```
In [3]: os.chdir('C:\\Users\\lenovo\\Desktop')
```

```
In [4]: df=pd.read_csv('CHD_preprocessed.csv')
```

```
In [5]: df.head()
```

```
Out[5]:
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate
0	1	39	1	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.1
1	0	46	0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.1
2	1	48	0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.1
3	0	61	1	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.1
4	0	46	1	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.1

```
In [6]: df.tail()
```

```
Out[6]:
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate
4128	1	50	0	1	1.0	0.0	0	1	0	313.0	179.0	92.0	25.97	
4129	1	51	1	1	43.0	0.0	0	0	0	207.0	126.5	80.0	19.71	
4130	0	48	0	1	20.0	0.0	0	0	0	248.0	131.0	72.0	22.00	
4131	0	44	0	1	15.0	0.0	0	0	0	210.0	126.5	87.0	19.16	
4132	0	52	0	0	0.0	0.0	0	0	0	269.0	133.5	83.0	21.47	

```
In [7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4133 entries, 0 to 4132
Data columns (total 16 columns):
#   Column              Non-Null Count  Dtype
---  -
0   male                 4133 non-null   int64
1   age                  4133 non-null   int64
2   education            4133 non-null   int64
3   currentSmoker        4133 non-null   int64
4   cigsPerDay           4133 non-null   float64
5   BPMeds               4133 non-null   float64
6   prevalentStroke       4133 non-null   int64
7   prevalentHyp         4133 non-null   int64
8   diabetes             4133 non-null   int64
9   totChol              4133 non-null   float64
10  sysBP                4133 non-null   float64
11  diaBP                4133 non-null   float64
12  BMI                  4133 non-null   float64
```

```
13 heartRate      4133 non-null float64
14 glucose        4133 non-null float64
15 TenYearCHD     4133 non-null int64
dtypes: float64(8), int64(8)
memory usage: 516.8 KB
```

```
In [8]: df.size
```

Out[8]: 66128

```
In [9]: df.shape
```

Out[9]: (4133, 16)

```
In [10]: df.isna().sum()
```

```
Out[10]: male      0
age      0
education  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
```

```
In [11]: df.describe()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol
count	4133.000000	4133.000000	4133.000000	4133.000000	4133.000000	4133.000000	4133.000000	4133.000000	4133.000000	4133.000000
mean	0.427293	49.557222	0.280668	0.494798	9.101621	0.034358	0.006049	0.311154	0.025647	236.664400
std	0.494745	8.561628	0.449380	0.500033	11.918440	0.182168	0.077548	0.463022	0.158100	43.909180
min	0.000000	32.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	107.000000
25%	0.000000	42.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	206.000000
50%	0.000000	49.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	234.000000
75%	1.000000	56.000000	1.000000	1.000000	20.000000	0.000000	0.000000	1.000000	0.000000	262.000000
max	1.000000	70.000000	1.000000	1.000000	70.000000	1.000000	1.000000	1.000000	1.000000	600.000000

```
In [14]: x = df.drop("TenYearCHD",axis=1)
y = df['TenYearCHD']
```

```
In [16]: x
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate
0	1	39	1	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	
1	0	46	0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	
2	1	48	0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	
3	0	61	1	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	
4	0	46	1	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	
...
4128	1	50	0	1	1.0	0.0	0	1	0	313.0	179.0	92.0	25.97	
4129	1	51	1	1	43.0	0.0	0	0	0	207.0	126.5	80.0	19.71	

4130	0	48	0	1	20.0	0.0	0	0	0	248.0	131.0	72.0	22.00
4131	0	44	0	1	15.0	0.0	0	0	0	210.0	126.5	87.0	19.16
4132	0	52	0	0	0.0	0.0	0	0	0	269.0	133.5	83.0	21.47

4133 rows × 15 columns

In [17]:

 y

Out[17]:

0	0
1	0
2	0
3	1
4	0

```

..
4128  1
4129  0
4130  0
4131  0
4132  0

```

Name: TenYearCHD, Length: 4133, dtype: int64

Train - Test Splitting

In [21]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

In [22]:

y_train

Out[22]:

173	1
1022	0
3182	0
331	1
2222	0

3444	0
466	0
3092	0
3772	0
860	0

```
Name: TenYearCHD, Length: 3306, dtype: int64
```

In [23]:

y_test

Out[23]:

1864	0
1210	0
1924	0
1752	0
1095	0

	..
881	0
25	1
3256	0
2269	0
1074	0

```
Name: TenYearCHD, Length: 827, dtype: int64
```

In [31]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression().fit(x_train,y_train)
model.score(x_train,y_train)
```

Out[31]:

0.8566243194192378

In [1]:

H = [1, 1, 1, 2, 3, 3, 4, 5, 6, 4, 4, 4, 5, 6, 6, 6, 7, 7, 8, 8, 9, 9, 9, 10, 10, 10, 10]

In [2]:

```
print(type(H))
```

<class 'list'>

In []:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js