In [ ]:
```
#Aim : To perform and find the accuracy of Support Vector Machine Algorithm i.e. SV
```

In [ ]:
```
# Name : Shruti Anil Dhote
# Roll no : 72
# Sec: C
# Subject : ET1
# Date :27/09/2024
```

In [1]:
```
import pandas as pd
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]:
```
import os
```

In [3]:
```
os.getcwd()
```

Out[3]:
```
'C:\\Users\\SURUTI DHOTE'
```

In [4]:
```
os.chdir("C:\\Users\\SURUTI DHOTE\\Desktop")
```

In [5]:
```
df=pd.read_csv("framingham.csv")
```

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

Out[6]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	dia
0	1	39	4.0	0	0.0	0.0	0	0	
1	0	46	2.0	0	0.0	0.0	0	0	
2	1	48	1.0	1	20.0	0.0	0	0	
3	0	61	3.0	1	30.0	0.0	0	1	
4	0	46	3.0	1	23.0	0.0	0	0	

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

Out[7]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalent
count	4240.000000	4240.000000	4135.000000	4240.000000	4211.000000	4187.000000	4240.0
mean	0.429245	49.580189	1.979444	0.494104	9.005937	0.029615	0.0
std	0.495027	8.572942	1.019791	0.500024	11.922462	0.169544	0.0
min	0.000000	32.000000	1.000000	0.000000	0.000000	0.000000	0.0
25%	0.000000	42.000000	1.000000	0.000000	0.000000	0.000000	0.0
50%	0.000000	49.000000	2.000000	0.000000	0.000000	0.000000	0.0
75%	1.000000	56.000000	3.000000	1.000000	20.000000	0.000000	0.0
max	1.000000	70.000000	4.000000	1.000000	70.000000	1.000000	1.0

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4240 entries, 0 to 4239
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             4240 non-null   int64  
 1   age              4240 non-null   int64  
 2   education        4135 non-null   float64
 3   currentSmoker    4240 non-null   int64  
 4   cigsPerDay       4211 non-null   float64
 5   BPMeds           4187 non-null   float64
 6   prevalentStroke  4240 non-null   int64  
 7   prevalentHyp     4240 non-null   int64  
 8   diabetes         4240 non-null   int64  
 9   totChol          4190 non-null   float64
 10  sysBP            4240 non-null   float64
 11  diaBP            4240 non-null   float64
 12  BMI              4221 non-null   float64
 13  heartRate        4239 non-null   float64
 14  glucose          3852 non-null   float64
 15  TenYearCHD       4240 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 530.1 KB
```

In [9]:
```
df.isna().sum()
```

Out[9]:

```
male                0
age                 0
education         105
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
```

In [10]:
```
#Since, only a few rows have null values in them, we are only removing those rows f
#df = df.dropna(subset=['heartRate','BMI','cigsPerDay','totChol','BPMeds'])
```

In [11]:
```
df
```

Out[11]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp
0	1	39	4.0	0	0.0	0.0	0	0
1	0	46	2.0	0	0.0	0.0	0	0
2	1	48	1.0	1	20.0	0.0	0	0
3	0	61	3.0	1	30.0	0.0	0	1
4	0	46	3.0	1	23.0	0.0	0	0
...	...	...	...	...	...	...	...	...
4235	0	48	2.0	1	20.0	NaN	0	0
4236	0	44	1.0	1	15.0	0.0	0	0
4237	0	52	2.0	0	0.0	0.0	0	0
4238	1	40	3.0	0	0.0	0.0	0	1
4239	0	39	3.0	1	30.0	0.0	0	0

4240 rows × 16 columns

Missing Value Treatment

In [12]:
```
df['glucose'].fillna(value = df['glucose'].mean(),inplace=True)
```

In [13]:
```
df['education'].fillna(value = df['education'].mean(),inplace=True)
```

In [14]:
```
df['heartRate'].fillna(value = df['heartRate'].mean(),inplace=True)
```

In [15]:
```
df['BMI'].fillna(value = df['BMI'].mean(),inplace=True)
```

In [16]:
```
df['cigsPerDay'].fillna(value = df['cigsPerDay'].mean(),inplace=True)
```

In [17]:
```
df['cigsPerDay'].fillna(value = df['cigsPerDay'].mean(),inplace=True)
```

In [18]:
```
df['BPMeds'].fillna(value = df['BPMeds'].mean(),inplace=True)
```

In [19]:
```
df.isna().sum()
```

Out[19]:
```
male               0
age                0
education          0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol           50
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
TenYearCHD         0
dtype: int64
```

In [20]:
```
#Splitting the dependent and independent variables.
x = df.drop("TenYearCHD",axis=1)
y = df['TenYearCHD']
```

In [21]:
```
x #checking the features
```

Out[21]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp
0	1	39	4.0	0	0.0	0.000000	0	0
1	0	46	2.0	0	0.0	0.000000	0	0
2	1	48	1.0	1	20.0	0.000000	0	0
3	0	61	3.0	1	30.0	0.000000	0	1
4	0	46	3.0	1	23.0	0.000000	0	0
...	...	...	...	...	...	...	...	...
4235	0	48	2.0	1	20.0	0.029615	0	0
4236	0	44	1.0	1	15.0	0.000000	0	0
4237	0	52	2.0	0	0.0	0.000000	0	0
4238	1	40	3.0	0	0.0	0.000000	0	1
4239	0	39	3.0	1	30.0	0.000000	0	0

4240 rows × 15 columns

Train Test Split

In [30]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

In [31]:
```
y_train
```

Out[31]:
```
1427    0
3257    0
3822    0
1263    0
3575    0
       ..
3444    0
466     0
3092    0
3772    0
860     0
Name: TenYearCHD, Length: 3392, dtype: int64
```

In [35]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

In [43]:
```
x_test = x_test.dropna()
y_test = y_test.loc[x_test.index]  # Ensure the target is aligned with x_test after
```

In [44]:
```
x_test = x_test.dropna()
y_test = y_test.loc[x_test.index]  # Ensure the target is aligned with x_test after
```

In [45]:
```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean')  # You can also use 'median', 'most_freque
x_test = imputer.fit_transform(x_test)
```

In [46]:
```
from sklearn.ensemble import HistGradientBoostingClassifier

classifier = HistGradientBoostingClassifier()
classifier.fit(x_train, y_train)
acc = classifier.score(x_test, y_test)
print(acc)
```

```
0.8430439952437574
```

In [ ]: