Heart Disease

Here is where your presentation begins

Comparative study of ML model to predict heart disease

In my project, we will analyze the most relevant/risk factors of heart disease as well as predict the overall risk using logistic regression. Since the early prognosis of cardiovascular diseases can aid in making decisions on lifestyle changes to reduce complications and save the life of high-risk patients. Where the World Health Organization has estimated 12 million deaths occur worldwide, every year due to heart diseases. And compare the most accuracy ML model to predict heart disease.

The question will disscuse about it in our data is:

- Whose is more susceptible to heart disease men or women?
- Is the Total cholesterol effect to increase the probability of susceptible CHD?
- Which is the most algorithm predict accurately?

The dataset is containe 15 features, and I divided it into 5 categorical to be clearer.
The features is:

-Demographic
1-Sex: male or female(Nominal)
2-Age: Age of the patient;(Continuous - Although the recorded ages have been truncated to whole numbers, the concept of age is continuous)

-Behavioral
3-Current Smoker: whether or not the patient is a current smoker (Nominal)
4-Cigs Per Day: the number of cigarettes that the person smoked on average in one day.(can be considered continuous as one can have any number of cigarettes, even half a cigarette.)

-Medical(history)
5-BP Meds: whether or not the patient was on blood pressure medication (Nominal)
6-Prevalent Stroke: whether or not the patient had previously had a stroke (Nominal)
7-Prevalent Hyp: whether or not the patient was hypertensive (Nominal)
8-Diabetes: whether or not the patient had diabetes (Nominal)

-Medical(current)
9-Tot Chol: total cholesterol level (Continuous)
10-Sys BP: systolic blood pressure (Continuous)
11-Dia BP: diastolic blood pressure (Continuous)
12-BMI: Body Mass Index (Continuous)
13-Heart Rate: heart rate (Continuous - In medical research, variables such as heart rate though in fact discrete, yet are considered continuous because of large number of possible values.)
14-Glucose: glucose level (Continuous)

-Predict variable (desired target)
15-TenYearCHD: 10-year risk of coronary heart disease CHD (binary: “1”, means “Yes”, “0” means “No”)

First: Import Libraries

In [1]:
```python
# loading dataset
import pandas as pd
import numpy as np
import seaborn as sns
import seaborn as sn
# visualisation
import matplotlib.pyplot as plt
%matplotlib inline
# Resampling imbalanced dataset
from sklearn.utils import resample
# data splitting
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
import scipy.stats as st
from collections import Counter
from statsmodels.tools import add_constant as add_constant
# data preprocessing
from sklearn.preprocessing import StandardScaler
# data modeling
from sklearn.metrics import confusion_matrix,accuracy_score,roc_curve,classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
# ensembling
from mlxtend.classifier import StackingCVClassifier
```

Second: Check out the Data

In [2]:
```python
HeartDisease_dataset= pd.read_csv('predict heart disease.csv')
```

In [3]:
```python
HeartDisease_dataset.drop(['education'],axis=1,inplace=True)
HeartDisease_dataset.head()
```

Out[3]:

	male	age	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCH
0	1	39	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	
1	0	46	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	
2	1	48	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	
3	0	61	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	
4	0	46	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	

In [4]:
```python
HeartDisease_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4238 entries, 0 to 4237
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   male             4238 non-null   int64
 1   age              4238 non-null   int64
 2   currentSmoker    4238 non-null   int64
 3   cigsPerDay       4209 non-null   float64
 4   BPMeds           4185 non-null   float64
 5   prevalentStroke  4238 non-null   int64
 6   prevalentHyp     4238 non-null   int64
 7   diabetes         4238 non-null   int64
 8   totChol          4188 non-null   float64
 9   sysBP            4238 non-null   float64
 10  diaBP            4238 non-null   float64
 11  BMI              4219 non-null   float64
 12  heartRate        4237 non-null   float64
 13  glucose          3850 non-null   float64
 14  TenYearCHD       4238 non-null   int64
dtypes: float64(8), int64(7)
memory usage: 496.8 KB
```

In [5]:
```python
HeartDisease_dataset.describe()
```

Out[5]:

	male	age	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	di
count	4238.000000	4238.000000	4238.000000	4209.000000	4185.000000	4238.000000	4238.000000	4238.000000	4188.000000	4238.000000	4238.000
mean	0.429212	49.584946	0.494101	9.003089	0.029630	0.005899	0.310524	0.025720	236.721585	132.352407	82.893
std	0.495022	8.572160	0.500024	11.920094	0.169584	0.076587	0.462763	0.158316	44.590334	22.038097	11.910
min	0.000000	32.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	107.000000	83.500000	48.000
25%	0.000000	42.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	206.000000	117.000000	75.000
50%	0.000000	49.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	234.000000	128.000000	82.000
75%	1.000000	56.000000	1.000000	20.000000	0.000000	0.000000	1.000000	0.000000	263.000000	144.000000	89.875
max	1.000000	70.000000	1.000000	70.000000	1.000000	1.000000	1.000000	1.000000	696.000000	295.000000	142.500

In [7]:
```python
HeartDisease_dataset.rename(columns={'male':'Sex_male'},inplace=True)
```

Third: Missing values

In [8]:
```
HeartDisease_dataset.isnull().sum()
```

Out[8]:
```
Sex_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
```

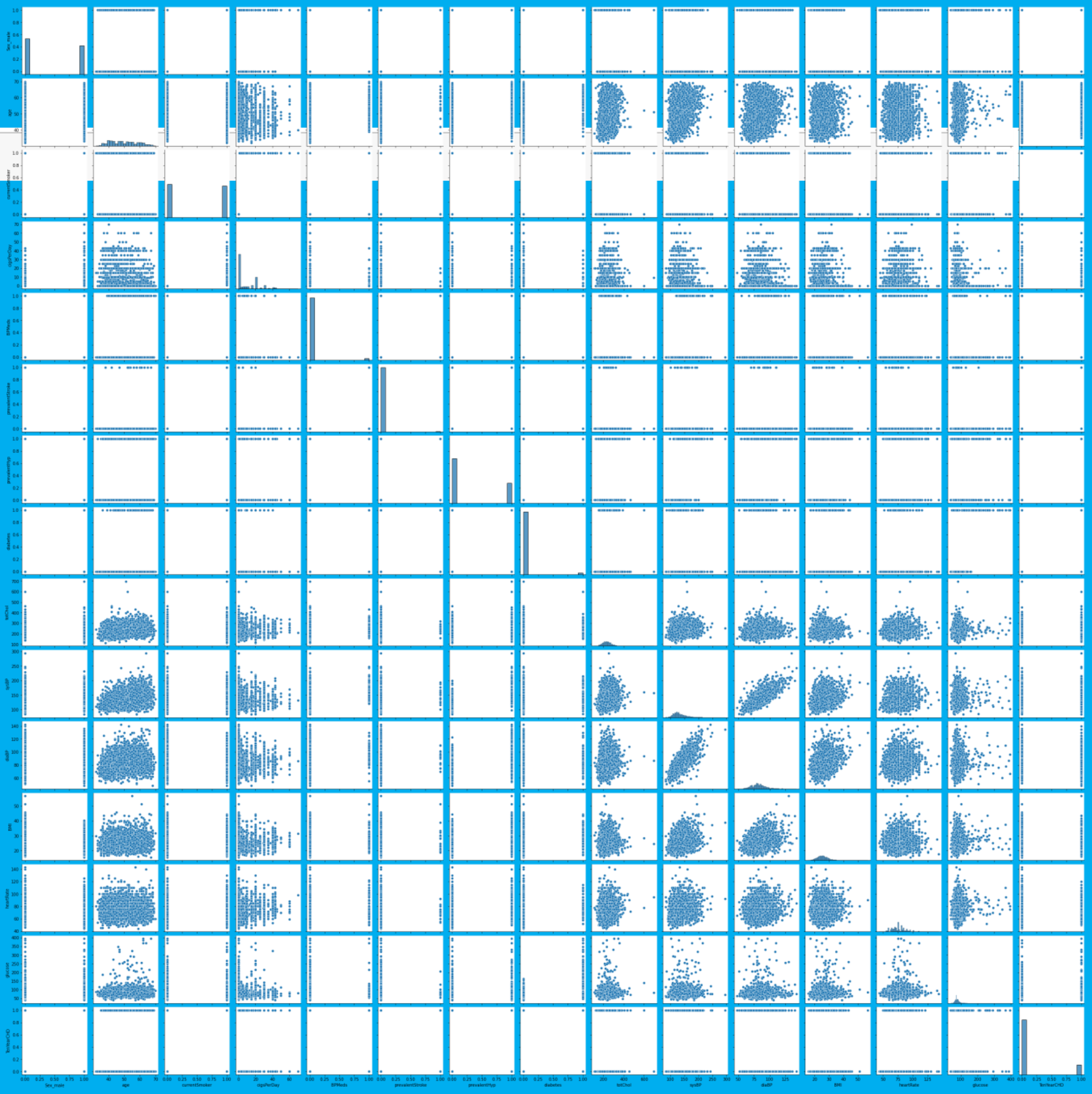
In [9]:
```
count=0
for i in HeartDisease_dataset.isnull().sum(axis=1):
    if i>0:
        count=count+1
print('Total number of rows with missing values is ', count)
print('since it is only',round((count/len(HeartDisease_dataset.index))*100), 'percent of the entire dataset the rows with mis
```

```
Total number of rows with missing values is  489
since it is only 12 percent of the entire dataset the rows with missing values are excluded.
```

In [10]:
```
HeartDisease_dataset.dropna(axis=0,inplace=True)
```

Fourth: Exploratory data analysis (EDA)

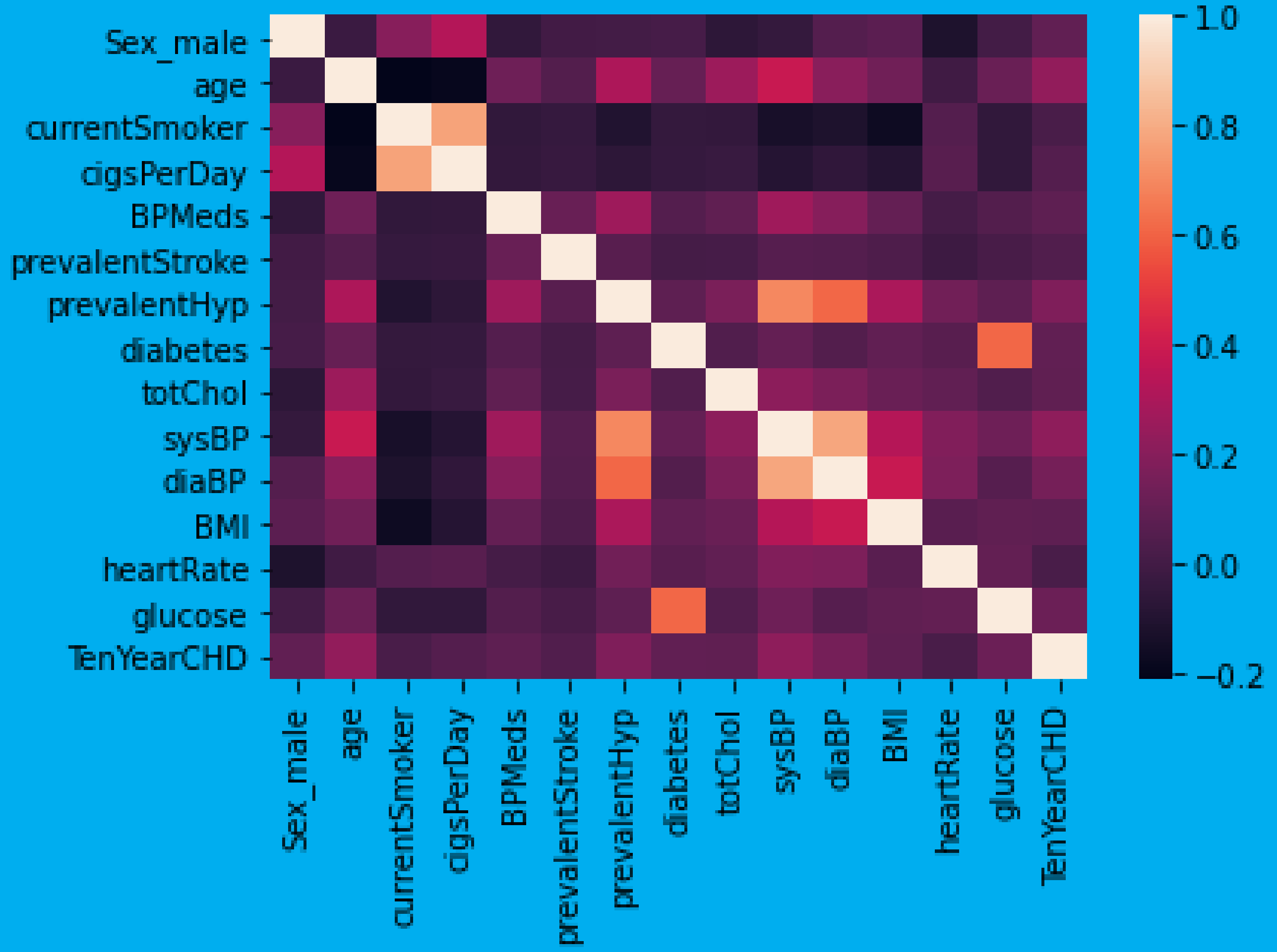
```
In [11]:  sns.pairplot(HeartDisease_dataset)
```



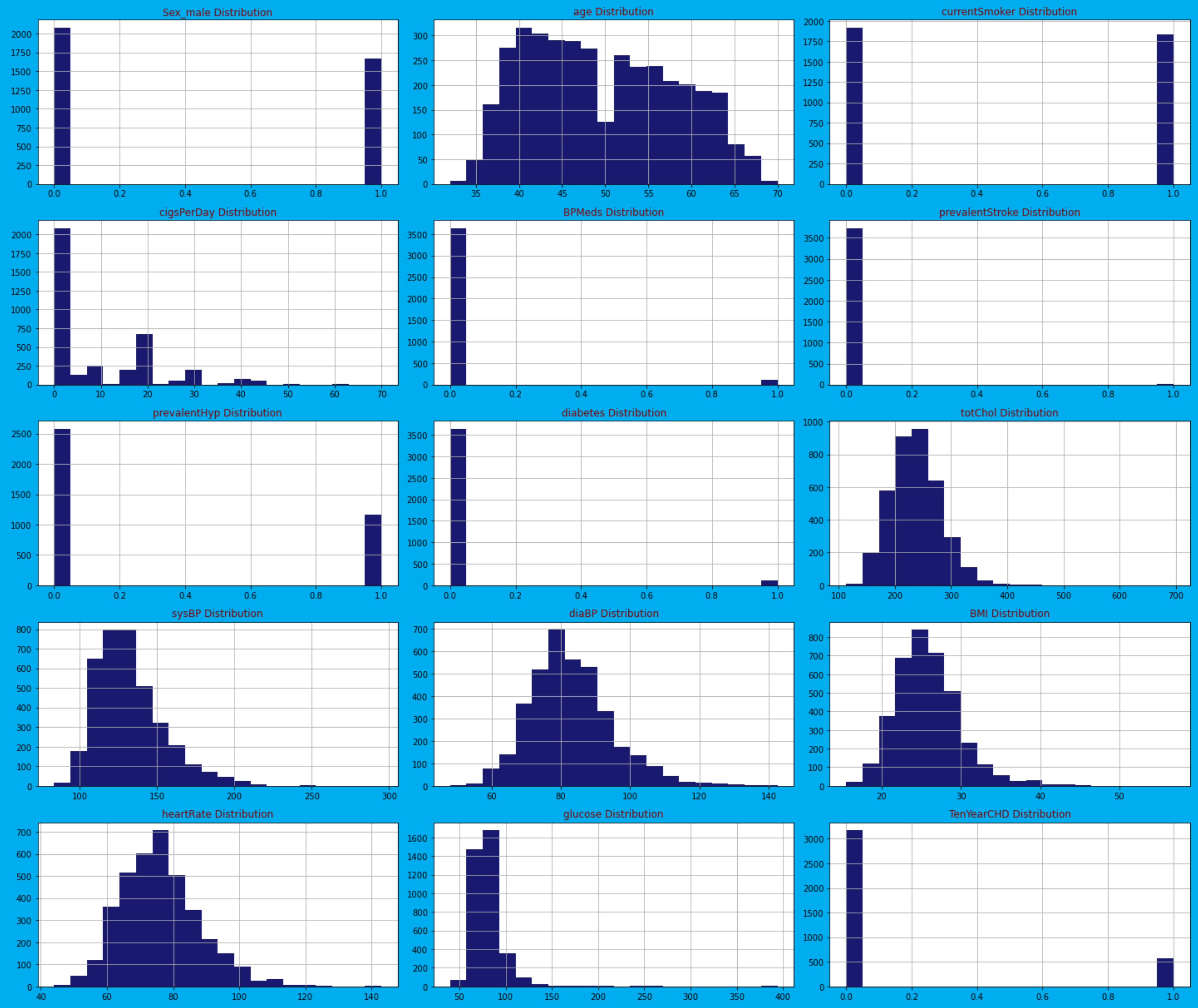
the two features [sysBP , diaBP] have a strong correlation.

In [12]:

```
sns.heatmap(HeartDisease_dataset.corr())
```

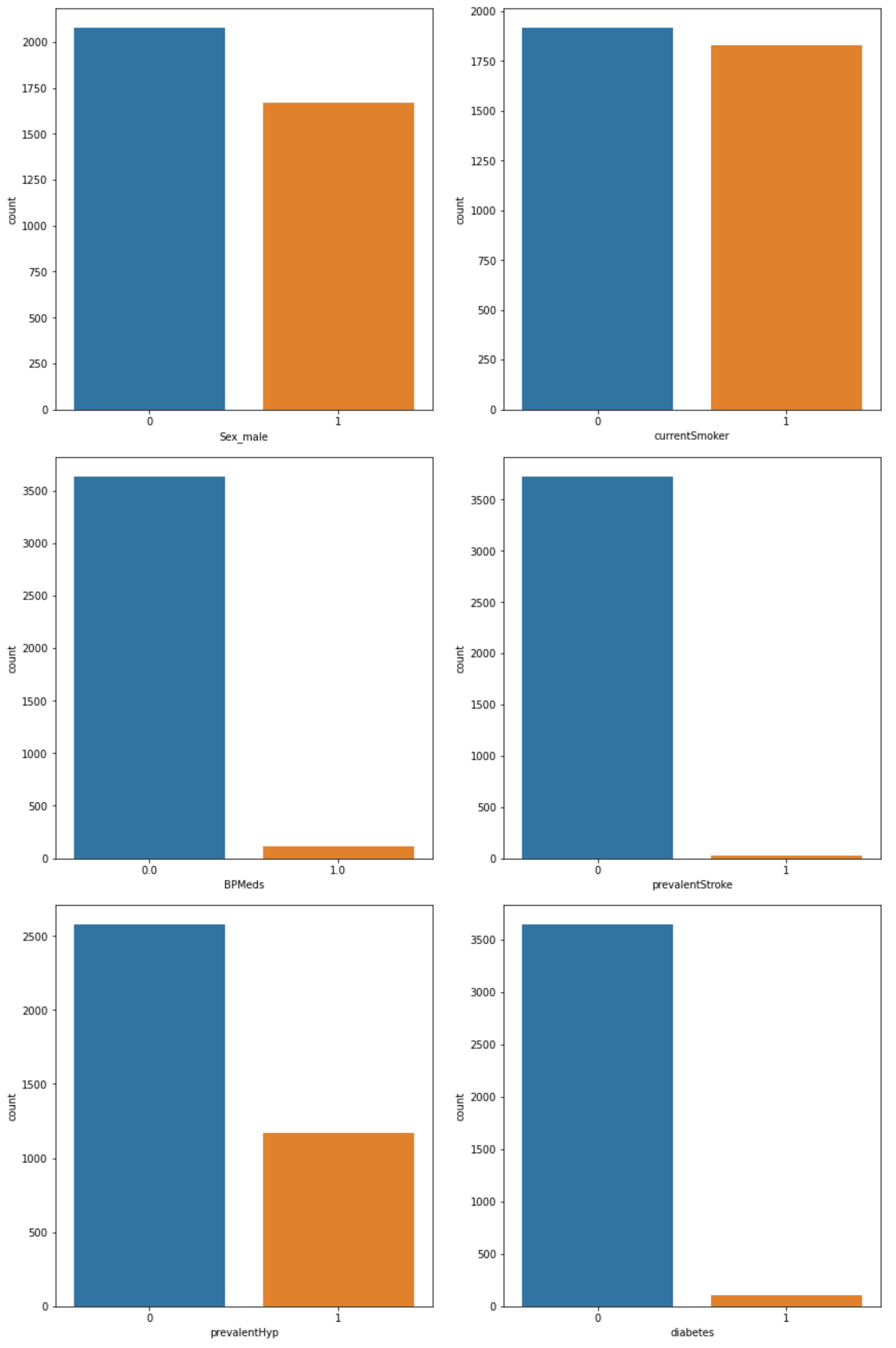


draw_histograms



Countplot

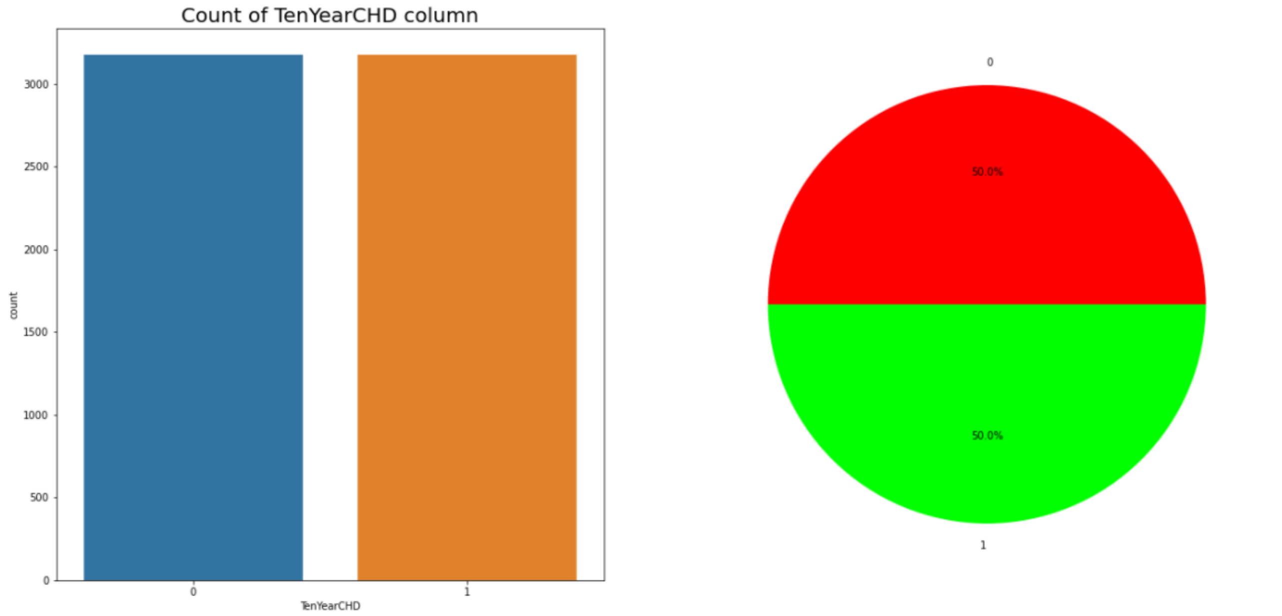
BPmeds, prevalentStroke and diabetes are highly imbalanced.
The number of Smokers and non-Smokers in currentSmoker is almost the same



Resampling imbalanced dataset by oversampling positive cases

```
In [17]: target1=HeartDisease_dataset[HeartDisease_dataset['TenYearCHD']==1]
         target0=HeartDisease_dataset[HeartDisease_dataset['TenYearCHD']==0]

In [18]: target1=resample(target1,replace=True,n_samples=len(target0),random_state=40)

In [19]: target=pd.concat([target0,target1])

In [20]: target['TenYearCHD'].value_counts()

Out[20]: 0    3177
         1    3177
         Name: TenYearCHD, dtype: int64
```

```
In [22]: #Distribution of heart disease cases in the balanced dataset, the outcome variable
         plt.figure(figsize=(12, 10), facecolor='w')
         plt.subplots_adjust(right=1.5)
         plt.subplot(121)
         sns.countplot(x="TenYearCHD", data=data)
         plt.title("Count of TenYearCHD column", size=20)
         plt.subplot(122)
         labels=[0,1]
         plt.pie(data["TenYearCHD"].value_counts(),autopct="%1.1f%%",labels=labels,colors=["red","lime"])
         plt.show()
```



Interpreting the results: Odds Ratio, Confidence Intervals and P Values

In [29]:

```
params = np.exp(result.params)
conf = np.exp(result.conf_int())
conf['OR'] = params
pvalue=round(result.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	0.000044	0.000274	0.000109	0.000
Sex_male	1.454877	2.198166	1.788313	0.000
age	1.054409	1.080897	1.067571	0.000
cigsPerDay	1.011730	1.028128	1.019896	0.000
totChol	1.000150	1.004386	1.002266	0.036
sysBP	1.013299	1.021791	1.017536	0.000
glucose	1.004343	1.010895	1.007614	0.000

- This fitted model shows that, holding all other features constant, the odds of getting diagnosed with heart disease for males (sex_male = 1)over that of females (sex_male = 0) is exp(0.5815) = 1.788313. In terms of percent change, we can say that the odds for males are 78.8% higher than the odds for females.

- We will see 7% increase in the odds of getting diagnosed with CDH for a one year increase in age since exp(0.0655) = 1.067571.

- We can see with every extra cigarette one smokes thers is a 2% increase in the odds of CDH.

- For Total cholosterol level and glucose level there is no significant change.

- There is a 1.7% increase in odds for every unit increase in systolic Blood Pressure.

Model Evaluation

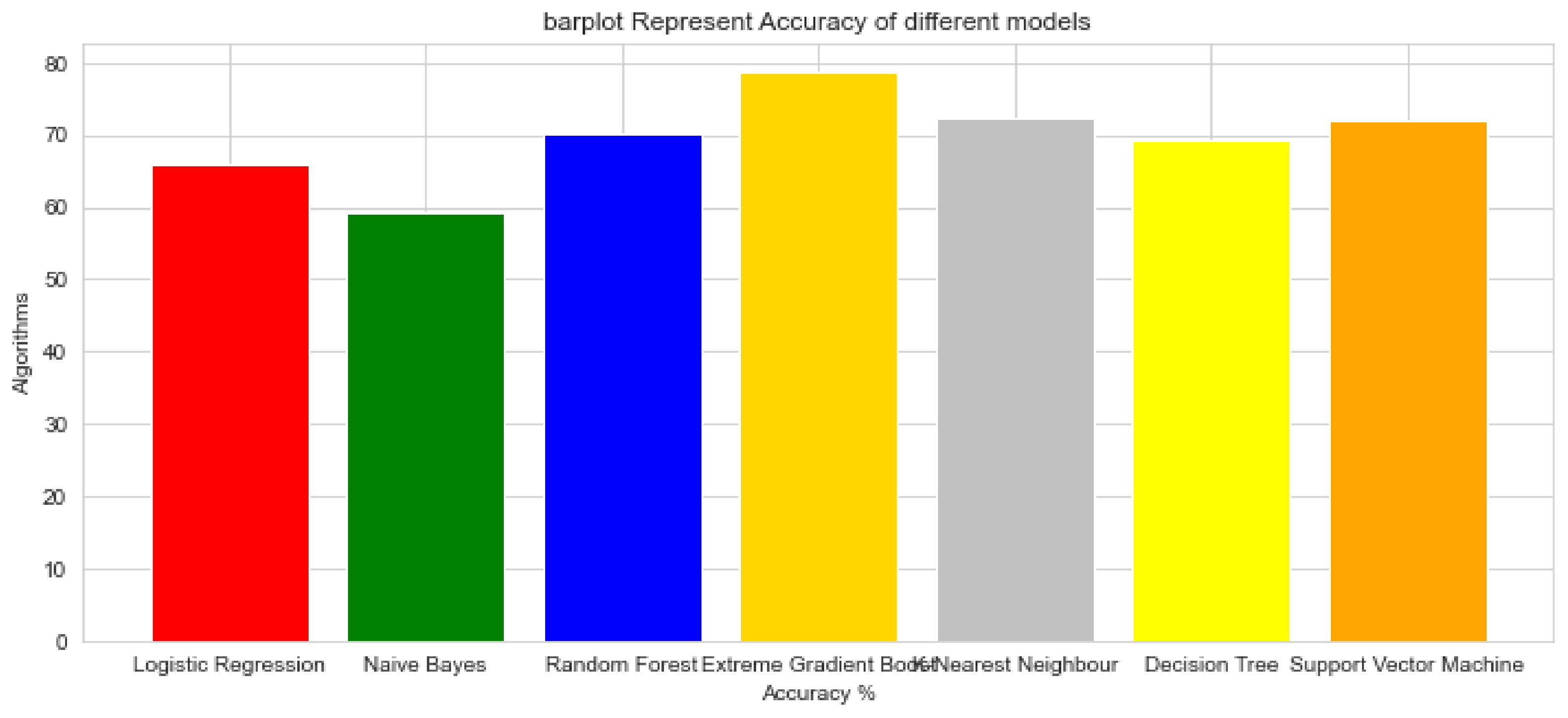
In [39]:

```
model_ev = pd.DataFrame({'Model': ['Logistic Regression','Naive Bayes','Random Forest','Extreme Gradient Boost',
                    'K-Nearest Neighbour','Decision Tree','Support Vector Machine'], 'Accuracy': [lr_acc_score*100,
                    nb_acc_score*100,rf_acc_score*100,xgb_acc_score*100,knn_acc_score*100,dt_acc_score*100,svc_acc_score*100]
model_ev
```

Out[39]:

	Model	Accuracy
0	Logistic Regression	66.089693
1	Naive Bayes	59.323367
2	Random Forest	70.180960
3	Extreme Gradient Boost	78.756884
4	K-Nearest Neighbour	72.462628
5	Decision Tree	69.236821
6	Support Vector Machine	71.990559

As we can see, the most accurate model is Extreme Gradient Boost.



Ensembling

In order to increase the accuracy of the model we use ensembling. Here we use stacking technique.

In [41]:

```
!pip install mlxtend
scv=StackingCVClassifier(classifiers=[xgb,knn,svc],meta_classifier= svc,random_state=42)
scv.fit(X_train,y_train)
scv_predicted = scv.predict(X_test)
scv_conf_matrix = confusion_matrix(y_test, scv_predicted)
scv_acc_score = accuracy_score(y_test, scv_predicted)
print("confussion matrix")
print(scv_conf_matrix)
print("\n")
print("Accuracy of StackingCVClassifier:",scv_acc_score*100,'\n')
print(classification_report(y_test,scv_predicted))
```

```
Accuracy of StackingCVClassifier: 78.83556254917387

              precision    recall  f1-score   support

           0       0.79      0.78      0.79       636
           1       0.78      0.80      0.79       635

    accuracy                           0.79      1271
   macro avg       0.79      0.79      0.79      1271
weighted avg       0.79      0.79      0.79      1271
```

End…

Thanks for all