Heart Risk Prediction Algorithm

Using Logistic Regression

A

ADM Course Project Report

in partial fulfilment of the degree

Bachelor of Technology
in
Computer Science & Engineering

By

Name: M Praveen	HTNo: 2303A1061
Name: Ch Adarsh	HTNo: 2303A1044
Name: B Sai Laxman	HTNo: 2303A1045
Name: B Pavan Adithya	HTNo: 2303A1047
Name: R Ajay	HTNo: 2303A51072

Under the guidance of

Bediga Sharan
Assistant Professor

Submitted to

School of Computer Science and Artificial Intelligence

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the **APPLICATIONS OF DATA MINING – Course Project** Report entitled **"HeartRisk Prediction Algorithm"** is a record of bonafide work carried out by the student(s) M.Praveen, Ch.Adrash, B.SaiLaxman, B.PavanAdithya, R.Ajay, bearing Hallticket No(s) 2303A51061,2303A51044,2303A51045,2303A51047,2303A51072 during the academic year 2024-25 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Computer Science & Engineering** by the SR University, Warangal.

Supervisor	**Head of the Department**
(Mr. Bediga Sharan)	(Dr. M. Sheshikala)
Assistant Professor	Professor

TABLE OF CONTENTS

---

1. OBJECTIVE OF THE PROJECT

Objective:

The objective of this project is to build a machine learning model, specifically a Logistic Regression model, to predict the 10-year risk of coronary heart disease (CHD) in individuals. This is achieved by analyzing various health-related features and demographic information from the Framingham Heart Study dataset. The project aims to identify key risk factors associated with CHD and develop a predictive model that can assist in early identification and prevention of heart disease.

Reasoning:

The code performs the following tasks:

1. **Data Loading and Preprocessing:** It loads the Framingham dataset, handles missing values, and addresses outliers.
2. **Exploratory Data Analysis (EDA):** It explores the dataset using visualizations and statistical analysis to identify patterns and relationships between variables.
3. **Feature Engineering:** It selects relevant features for the model, such as age, gender, smoking habits, cholesterol levels, blood pressure, and glucose levels.
4. **Model Building:** It trains a Logistic Regression model using the selected features.
5. **Model Evaluation:** It evaluates the model's performance using metrics such as accuracy and confusion matrix.

These steps indicate the goal of building and evaluating a predictive model for 10-year CHD risk

2. DEFINITIONS OF THE ELEMENTS USED IN THE PROJECT

1 . Dataset:

- **Framingham Heart Study Dataset:** A publicly available dataset used for cardiovascular research. It contains information about participants' health and demographic factors, including medical history, lifestyle choices, and lab results.

2. Libraries:

- **Pandas (pd):** Used for data manipulation and analysis, providing data structures like DataFrames.
- **NumPy (np):** Used for numerical computations, particularly for handling arrays and matrices.
- **Seaborn (sns):** Used for creating statistical visualizations, providing an interface to Matplotlib.
- **Matplotlib (plt):** Used for generating plots and charts, enabling visualization of data trends.
- **Scikit-learn (sklearn):** A machine learning library used for model building, evaluation, and data preprocessing.
    - **LogisticRegression:** A classification algorithm for predicting categorical outcomes.
    - **LabelEncoder:** Used for encoding categorical features into numerical values.
    - **accuracy_score, precision_score, confusion_matrix, classification_report:** Metrics for evaluating model performance.
    - **train_test_split:** Used to split data into training and testing sets.
    - **StandardScaler, MinMaxScaler, RobustScaler:** Used for feature scaling.

3. Variables:

- **male:** Gender of the participant (1 for male, 0 for female).
- **age:** Age of the participant in years.
- **education:** Education level of the participant (categorical).
- **currentSmoker:** Whether the participant is a current smoker (1 for yes, 0 for no).
- **cigsPerDay:** Number of cigarettes smoked per day.
- **BPMeds:** Whether the participant is taking blood pressure medication (1 for yes, 0 for no).
- **prevalentStroke:** Whether the participant has had a stroke (1 for yes, 0 for no).
- **prevalentHyp:** Whether the participant has prevalent hypertension (1 for yes, 0 for no).
- **diabetes:** Whether the participant has diabetes (1 for yes, 0 for no).
- **totChol:** Total cholesterol level.
- **sysBP:** Systolic blood pressure.
- **diaBP:** Diastolic blood pressure.
- **BMI:** Body Mass Index.
- **heartRate:** Heart rate.
- **glucose:** Glucose level.
- **TenYearCHD:** The target variable, indicating whether the participant developed CHD within 10 years (1 for yes, 0 for no).

4. Key Concepts:

- **Logistic Regression:** A statistical model used for binary classification, predicting the probability of an event occurring.
- **Correlation:** A measure of the relationship between two or more variables.
- **Outliers:** Data points that significantly differ from other observations.
- **EDA:** Exploratory Data Analysis, used to understand and summarize data before modeling.
- **Feature Engineering:** The process of selecting, transforming, and creating features to improve model performance.

5. Model Evaluation Metrics:

- **Accuracy:** The proportion of correctly classified instances.
- **Confusion Matrix:** A table showing the performance of a classification model by displaying the counts of true positive, true negative, false positive, and false negative predictions.

3. IMPLEMENTATION

1. Data Loading and Preprocessing:

- **Import necessary libraries:** pandas, numpy, seaborn, matplotlib, and sklearn modules are imported for data handling, visualization, and model building.
- **Load the dataset:** The Framingham dataset is loaded into a pandas DataFrame using pd.read_csv().
- **Handle missing values:** Missing values in columns like education, cigsPerDay, BPMeds, totChol, BMI, glucose, and heartRate are filled with their respective medians using fillna().
- **Address outliers:** Outliers in columns like cigsPerDay, totChol, sysBP, diaBP, BMI, heartRate, and glucose are treated using the interquartile range (IQR) method. Values outside the lower and upper fences are replaced with the fence values.

2. Exploratory Data Analysis (EDA):

- **Descriptive statistics:** The code calculates and displays descriptive statistics of the dataset using describe().
- **Univariate analysis:** It explores the distribution of individual variables using histograms, box plots, and count plots.
- **Bivariate analysis:** It investigates relationships between variables using scatter plots, line plots, and bar plots.
- **Correlation analysis:** It computes and visualizes the correlation matrix to identify potential relationships between features.

3. Feature Engineering:

- **Feature selection:** Relevant features for predicting CHD are selected based on EDA and domain knowledge. The selected features are: male, age, cigsPerDay, totChol, sysBP, diaBP, heartRate, and glucose.
- **Data splitting:** The dataset is split into training and testing sets using train_test_split() to evaluate the model's performance on unseen data.

4. Model Building:

- **Logistic Regression:** A Logistic Regression model is instantiated and trained using the training data (x_train, y_train) with log_reg.fit().

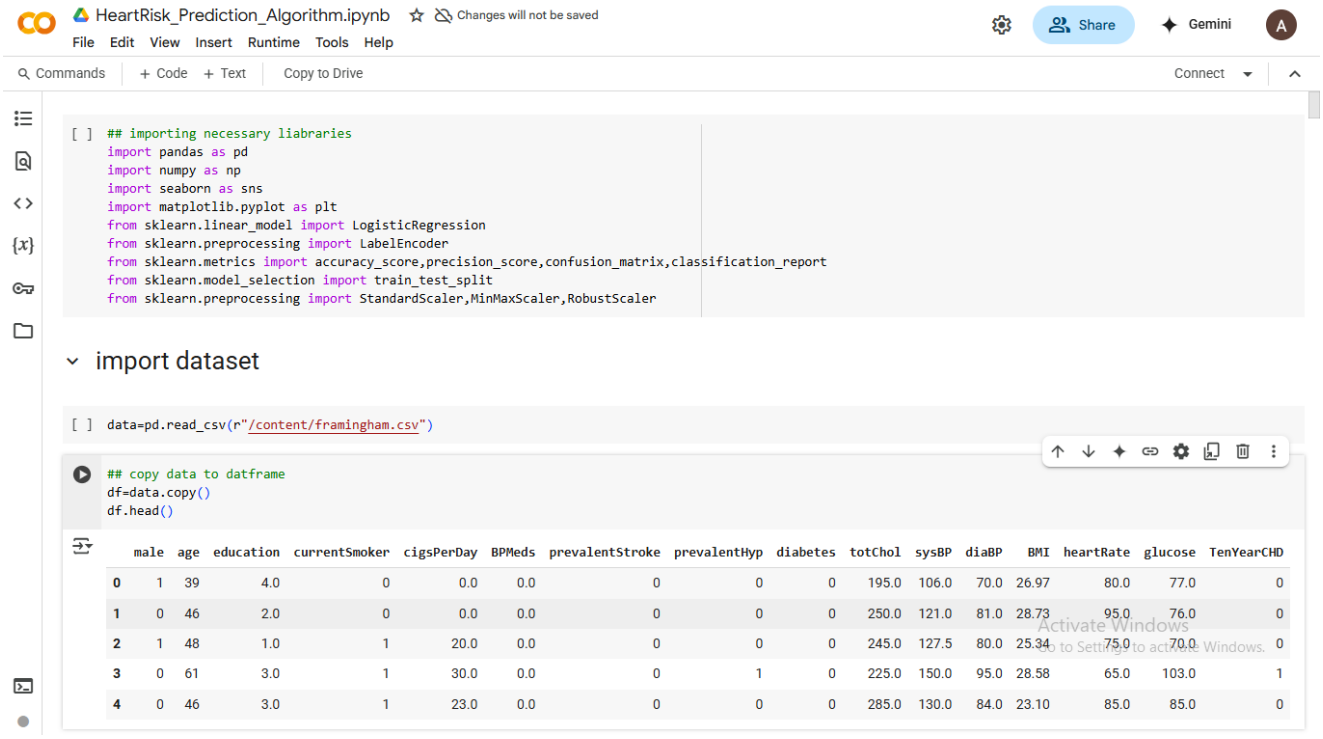
5. Model Evaluation:

- **Accuracy:** The accuracy of the model is calculated using accuracy_score().
- **Confusion Matrix:** The confusion matrix is generated using confusion_matrix() to assess the model's performance in terms of true positives, true negatives, false positives, and false negatives.

Overall:

The project implements a data science workflow to build a Logistic Regression model for predicting 10-year CHD risk. It involves data preprocessing, EDA, feature engineering, model training, and evaluation using relevant metrics. The results provide insights into the factors associated with CHD and the model's predictive performance

4. RESULT SCREENS



HeartRisk_Prediction_Algorithm.ipynb ☆ Changes will not be saved

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Copy to Drive Connect

```
4  0  46  3.0  1  23.0  0.0  0  0  0  285.0  130.0  84.0  23.1
```

```
## bottom five rows
df.tail()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
4233	1	50	1.0	1	1.0	0.0	0	1	0	313.0	179.0	92.0	25.97	66.0	86.0	1
4234	1	51	3.0	1	43.0	0.0	0	0	0	207.0	126.5	80.0	19.71	65.0	68.0	0
4235	0	48	2.0	1	20.0	NaN	0	0	0	248.0	131.0	72.0	22.00	84.0	86.0	0
4236	0	44	1.0	1	15.0	0.0	0	0	0	210.0	126.5	87.0	19.16	86.0	NaN	0
4237	0	52	2.0	0	0.0	0.0	0	0	0	269.0	133.5	83.0	21.47	80.0	107.0	0

```
## index of dataset
df.index
```

```
RangeIndex(start=0, stop=4238, step=1)
```

```
## shape of dataset
print("Total rows are - ",df.shape[0])
print("Total columns are - ",df.shape[1])
```

```
Total rows are -  4238
Total columns are -  16
```


datatype of dataset features

HeartRisk_Prediction_Algorithm.ipynb ☆ Changes will not be saved

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Copy to Drive Connect

```
df.dtypes
```

	0
male	int64
age	int64
education	float64
currentSmoker	int64
cigsPerDay	float64
BPMeds	float64
prevalentStroke	int64
prevalentHyp	int64
diabetes	int64
totChol	float64
sysBP	float64
diaBP	float64
BMI	float64
heartRate	float64
glucose	float64
TenYearCHD	int64

dtype: object

HeartRisk_Prediction_Algorithm.ipynb ☆ Changes will not be saved

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Copy to Drive Connect

```
[ ] ## information of dataset
    df.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
```

unique values in dataset


```
[ ] df.nunique()
```

	0

HeartRisk_Prediction_Algorithm.ipynb ☆ Changes will not be saved

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Copy to Drive Connect

[ ] df.nunique()

	0
male	2
age	39
education	4
currentSmoker	2
cigsPerDay	33
BPMeds	2
prevalentStroke	2
prevalentHyp	2
diabetes	2
totChol	248
sysBP	234
diaBP	146
BMI	1363
heartRate	73
glucose	143
TenYearCHD	2

dtype: int64

Activate Windows
Go to Settings to activate Windows.

Commands + Code + Text Changes will not be saved Connect

null values in dataset

```
df.isna().any()
```

	0
male	False
age	False
education	True
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

Commands + Code + Text Changes will not be saved Connect

```
df.isna().sum()
```

	0
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

Activate Windows
Go to Settings to activate Windows.

all null values columns are int or float datatype, so i will fill null values with median.

Commands + Code + Text Changes will not be saved Connect

check median of null values column

```
print(df['education'].median())
print(df['cigsPerDay'].median())
print(df['BPMeds'].median())
print(df['totChol'].median())
print(df['BMI'].median())
print(df['glucose'].median())
print(df['heartRate'].median())
```

```
2.0
0.0
0.0
234.0
25.4
78.0
75.0
```

fill NAN with its median values

```
df = df.copy()  # Make sure df is not a view

df['education'] = df['education'].fillna(2)
df['cigsPerDay'] = df['cigsPerDay'].fillna(0)
df['BPMeds'] = df['BPMeds'].fillna(0)
df['totChol'] = df['totChol'].fillna(234)
df['BMI'] = df['BMI'].fillna(25.4)
df['glucose'] = df['glucose'].fillna(78)
df['heartRate'] = df['heartRate'].fillna(75)
```



```
## now verify NAN in dataset
df.isna().sum()
```

Commands + Code + Text Changes will not be saved Connect

	0
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

statistical details of dataset

```
df.describe()
```

Commands + Code + Text Changes will not be saved Connect

```
df.describe()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	dia
count	4238.000000	4238.000000	4238.000000	4238.000000	4238.000000	4238.000000	4238.000000	4238.000000	4238.000000	4238.000000	4238.000000	4238.0000
mean	0.429212	49.584946	1.979471	0.494101	8.941482	0.029259	0.005899	0.310524	0.025720	236.689476	132.352407	82.8934
std	0.495022	8.572160	1.007081	0.500024	11.902399	0.168552	0.076587	0.462763	0.158316	44.327427	22.038097	11.9108
min	0.000000	32.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	107.000000	83.500000	48.0000
25%	0.000000	42.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	206.000000	117.000000	75.0000
50%	0.000000	49.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	234.000000	128.000000	82.0000
75%	1.000000	56.000000	3.000000	1.000000	20.000000	0.000000	0.000000	1.000000	0.000000	262.000000	144.000000	89.8750
max	1.000000	70.000000	4.000000	1.000000	70.000000	1.000000	1.000000	1.000000	1.000000	696.000000	295.000000	142.5000

```
##columns in dataset
col=df.columns
col
```

```
Index(['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

outliers in dataset


```
for i in col:
    plt.figure()
    sns.boxplot(x=df[i])
```

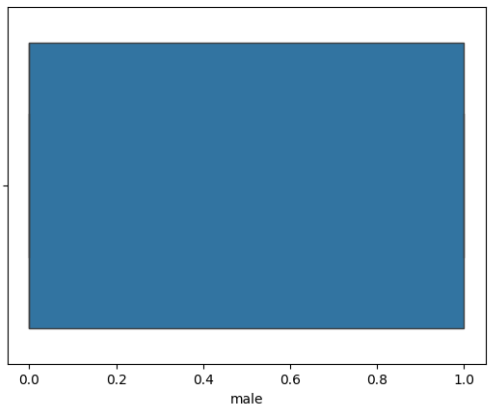
Commands + Code + Text Changes will not be saved Connect

col

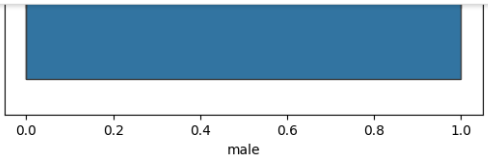
Index(['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
dtype='object')

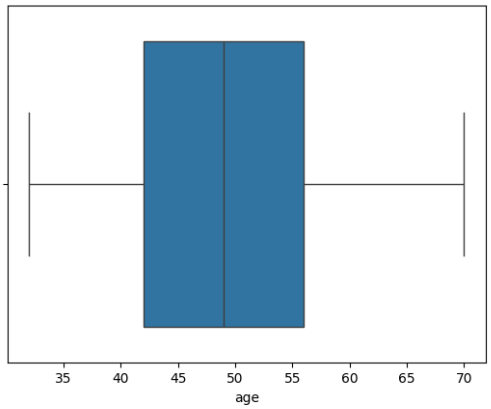
outliers in dataset

for i in col:
plt.figure()
sns.boxplot(x=df[i])

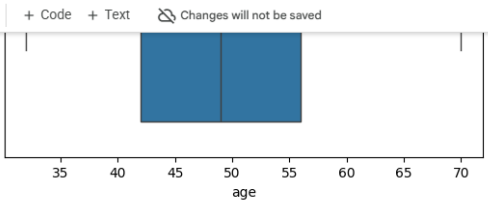


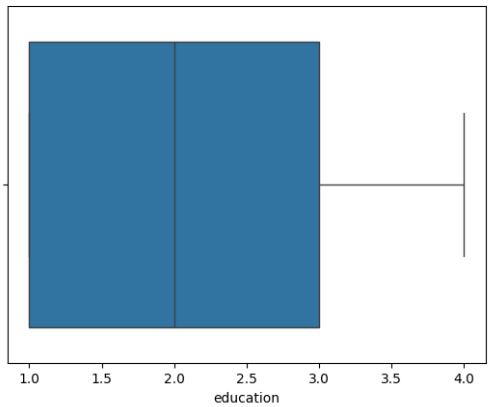
Activate Windows
Go to Settings to activate Windows.



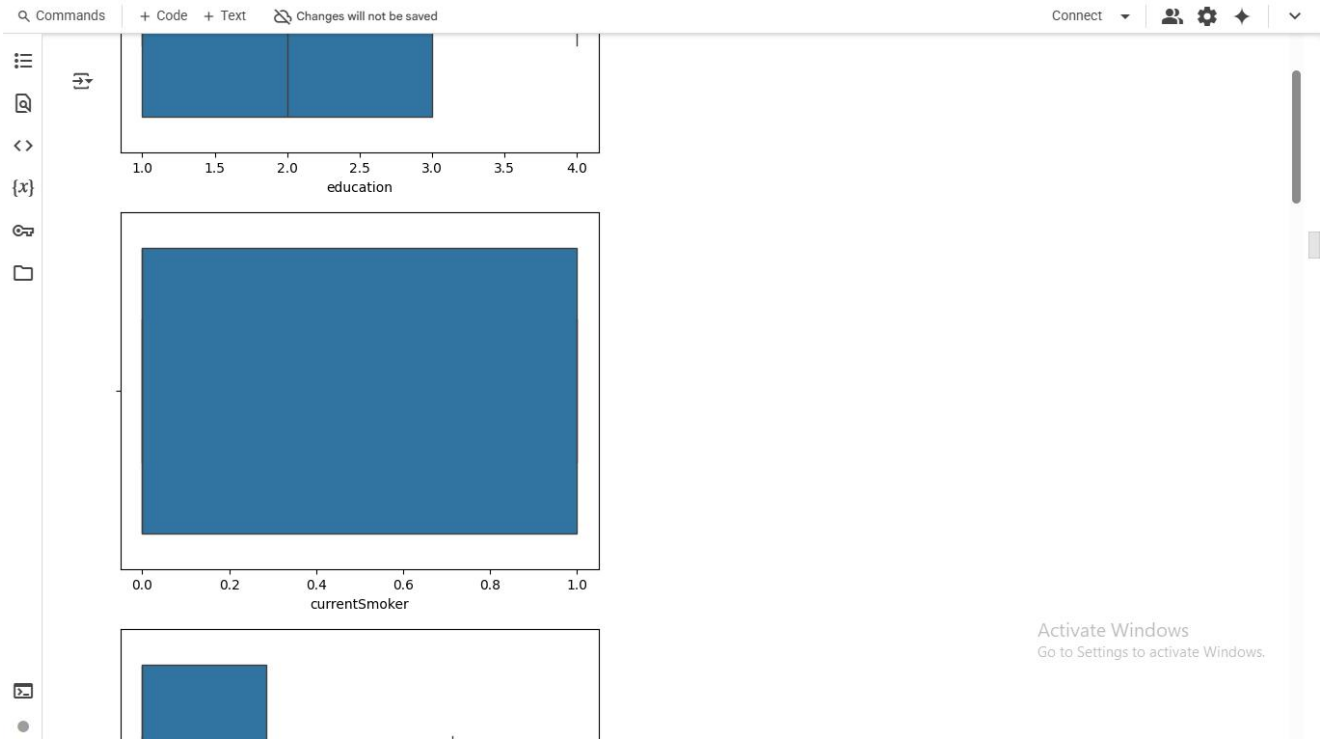


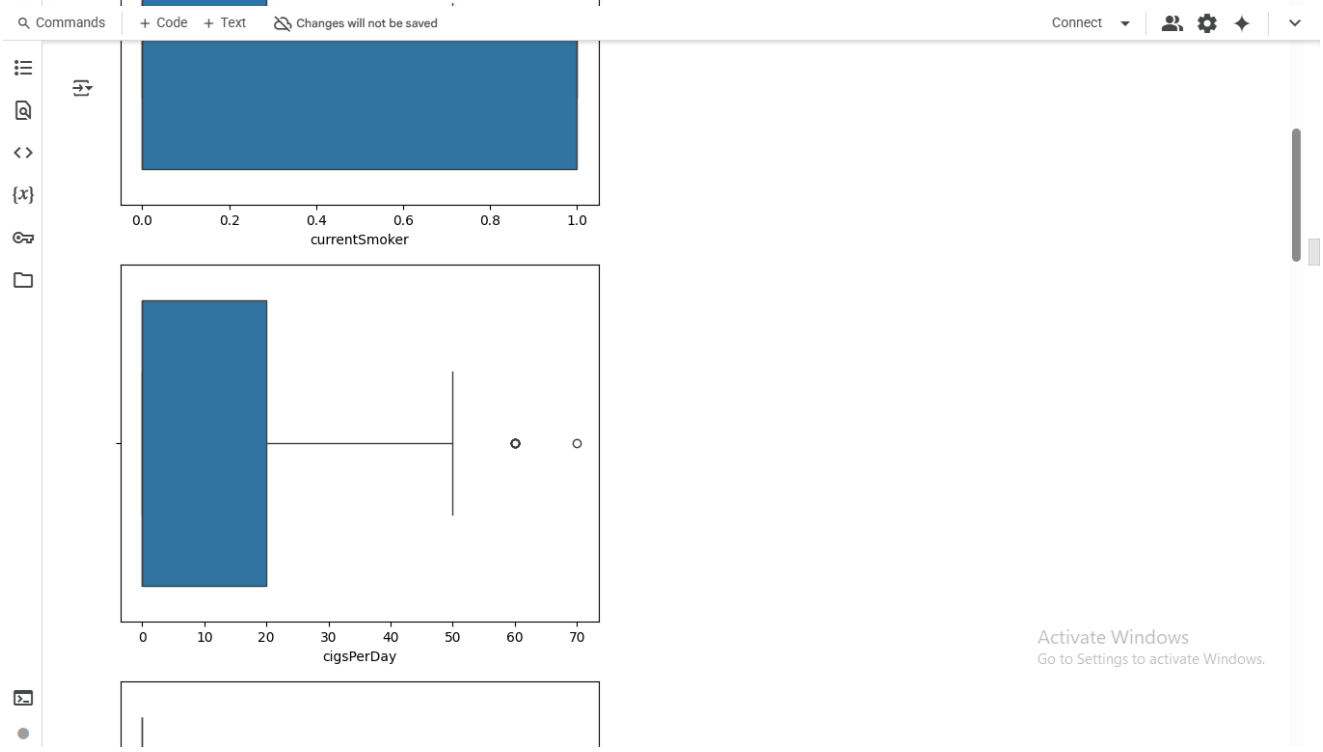
Commands + Code + Text Changes will not be saved Connect

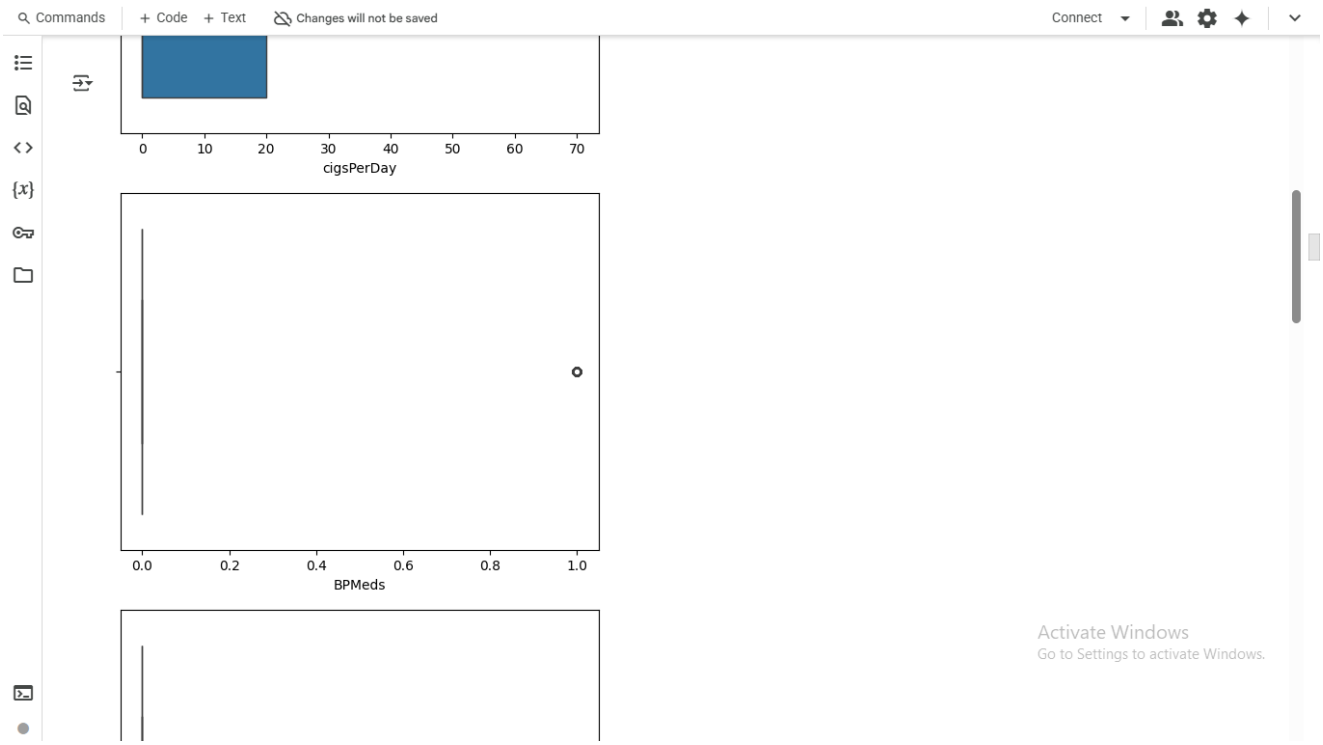


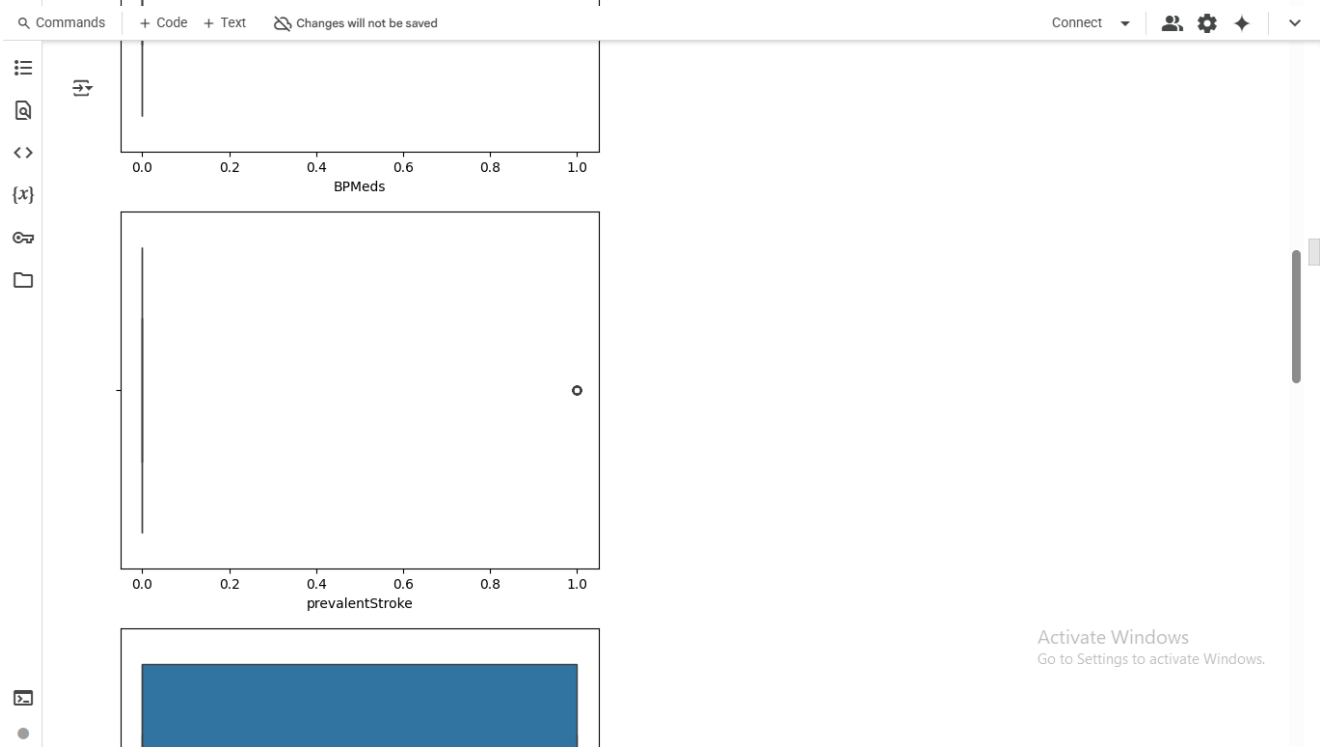


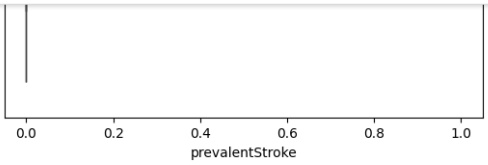
Activate Windows
Go to Settings to activate Windows.

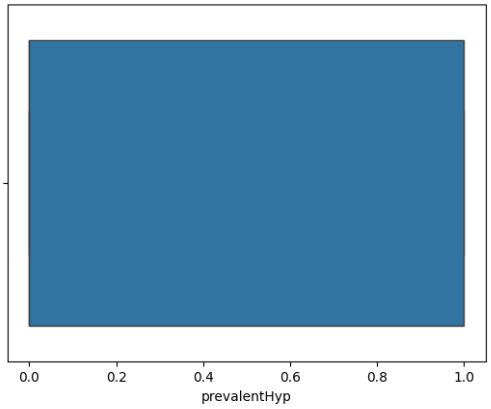




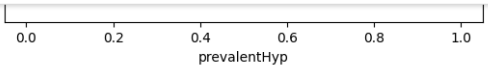


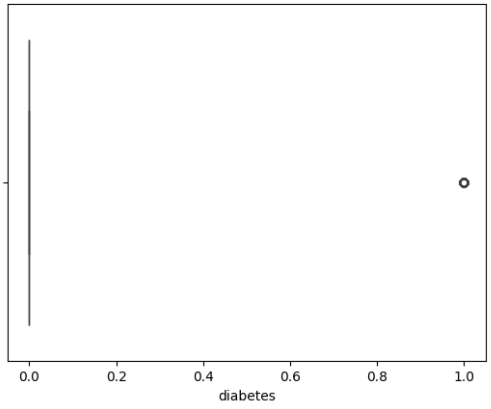


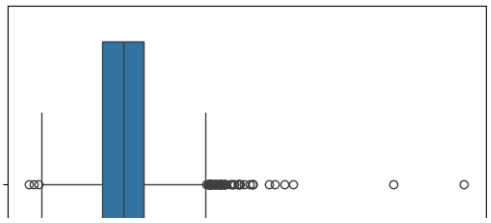


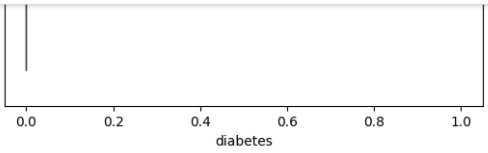


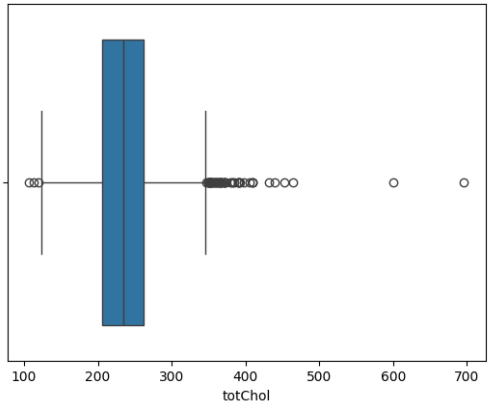
Commands + Code + Text Changes will not be saved Connect

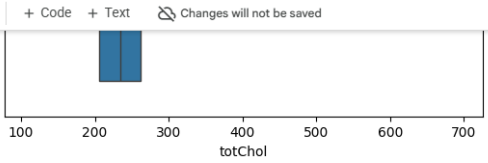


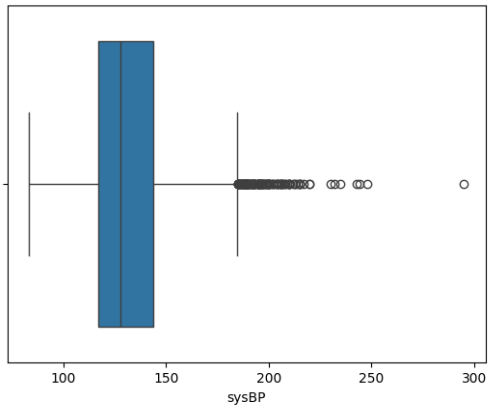


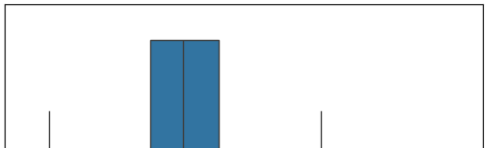


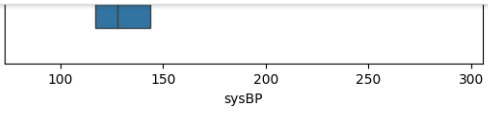


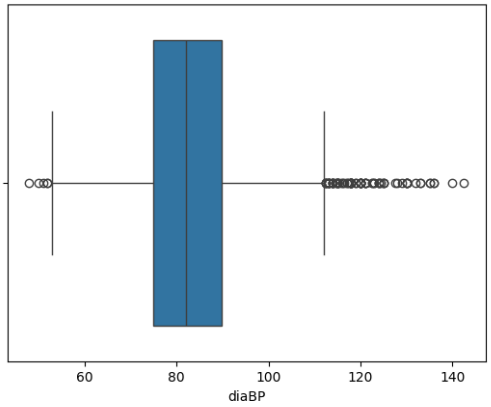


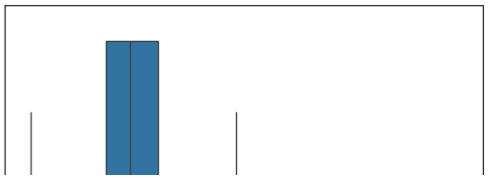


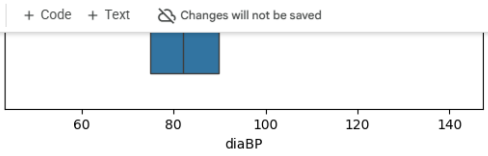


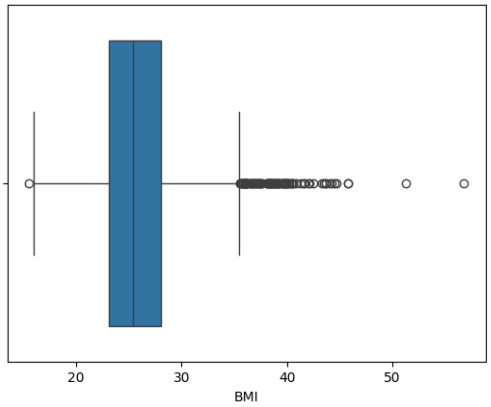


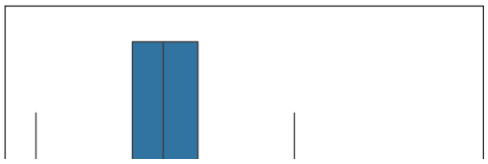


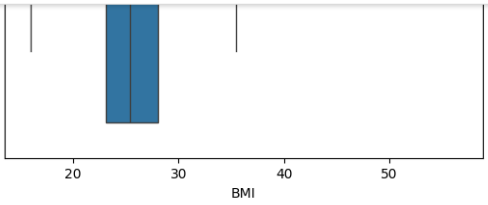


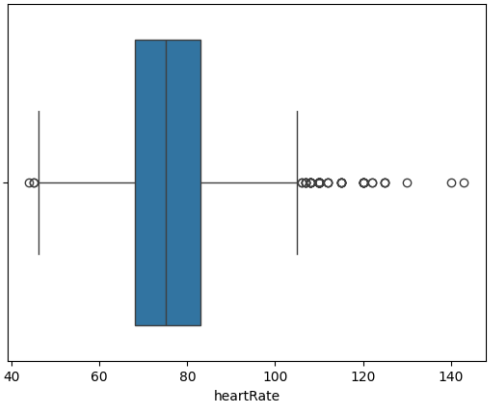


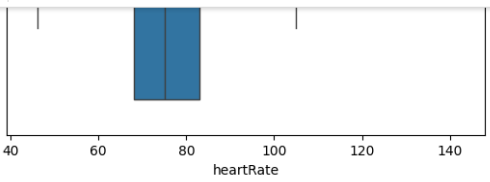


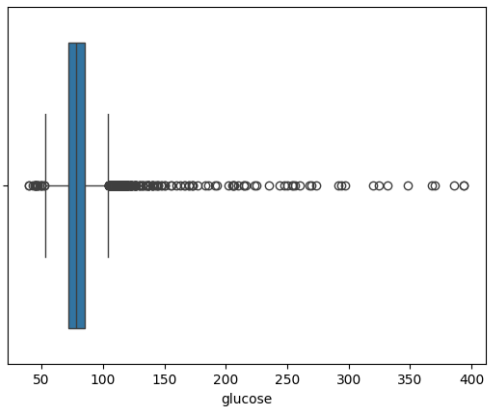


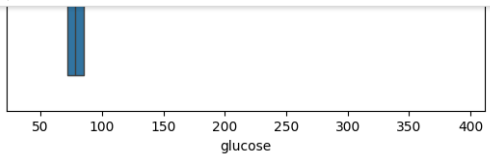


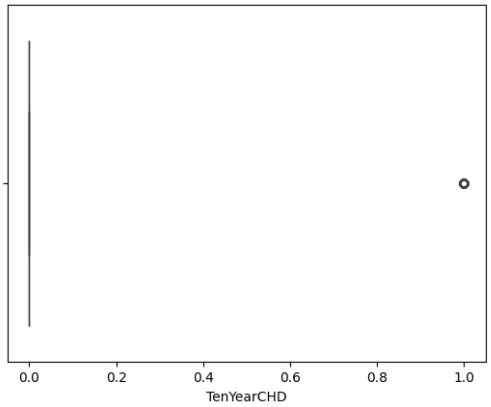








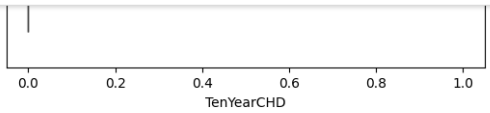




process on outliers

```
[ ] col=df[['cigsPerDay','totChol','sysBP','diaBP','BMI','heartRate','glucose']]
```

Commands + Code + Text Changes will not be saved Connect



process on outliers

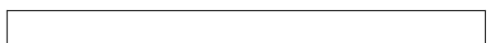
[ ] col=df[['cigsPerDay','totChol','sysBP','diaBP','BMI','heartRate','glucose']]

```
[ ] for i in col:
        q1=np.percentile(df[i],25)
        q3=np.percentile(df[i],75)
        iqr=q3-q1
        lower_fense=q1-1.5*(iqr)
        higher_fense=q3+1.5*(iqr)
        df[i]=np.where(df[i]<lower_fense,lower_fense,df[i])
        df[i]=np.where(df[i]>higher_fense,higher_fense,df[i])
```

```
[ ] column=df.columns
    column
```

```
Index(['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

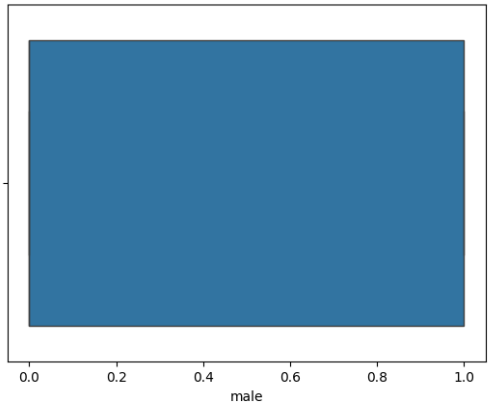
```
[ ] for p in column:
        plt.figure()
        sns.boxplot(x=df[p])
```

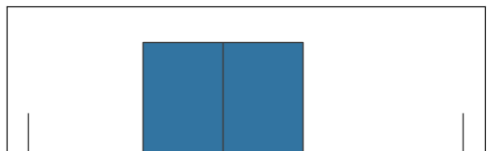



Commands + Code + Text Changes will not be saved Connect

```
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

```
for p in column:
    plt.figure()
    sns.boxplot(x=df[p])
```



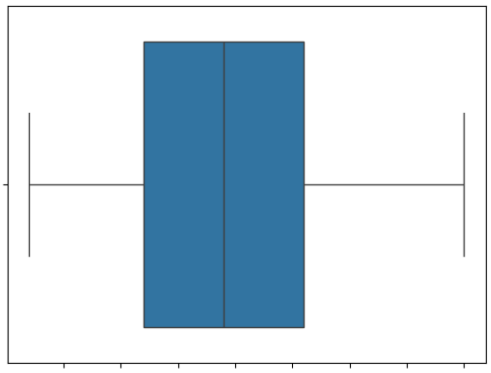


Commands + Code + Text Changes will not be saved Connect

```
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

```
for p in column:
    plt.figure()
    sns.boxplot(x=df[p])
```

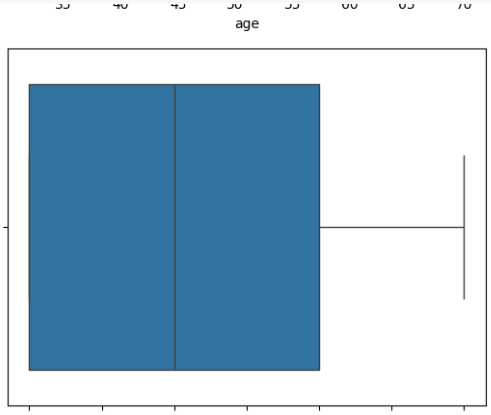
0.0
0.2
0.4
0.6
0.8
1.0
male



Commands + Code + Text Changes will not be saved Connect

```
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

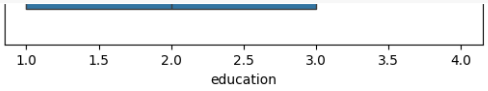
```
for p in column:
    plt.figure()
    sns.boxplot(x=df[p])
```

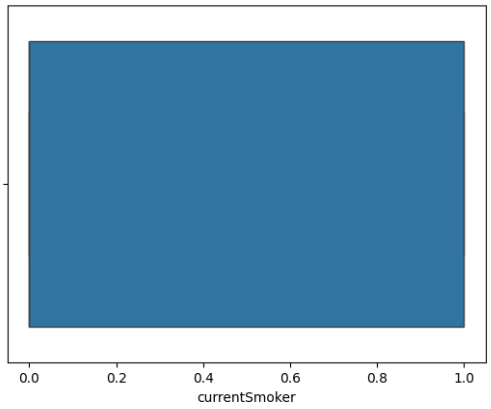


Commands + Code + Text Changes will not be saved Connect

```
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

```
for p in column:
    plt.figure()
    sns.boxplot(x=df[p])
```

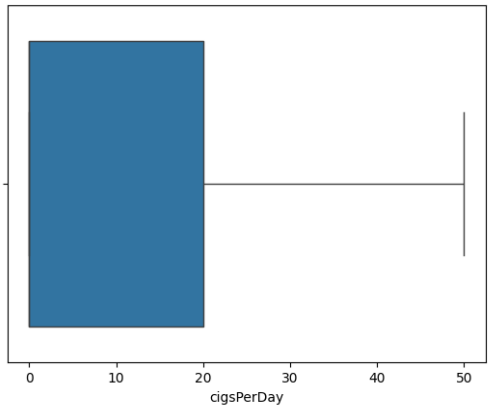




Commands + Code + Text Changes will not be saved Connect

```
      'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
     dtype='object')
```

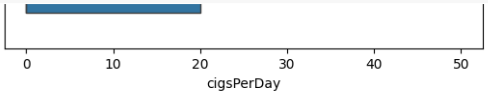
```
for p in column:
    plt.figure()
    sns.boxplot(x=df[p])
```

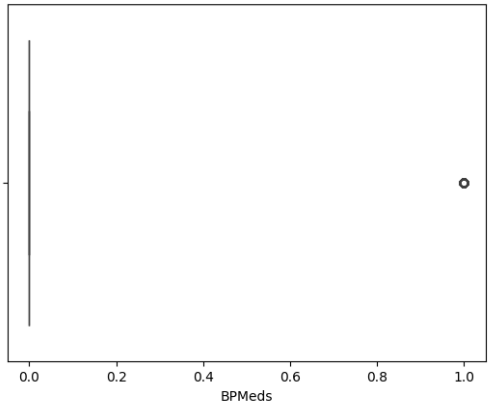


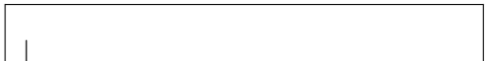
Commands + Code + Text Changes will not be saved Connect

'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
dtype='object')

for p in column:
plt.figure()
sns.boxplot(x=df[p])



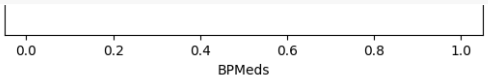


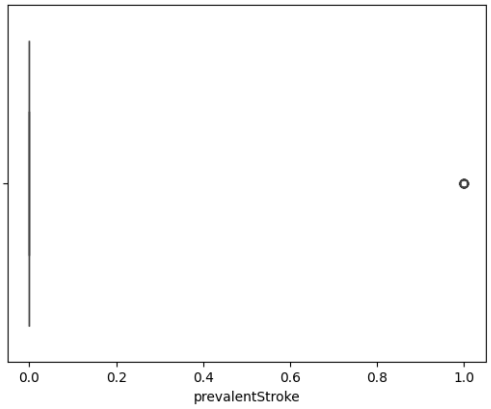


Commands + Code + Text Changes will not be saved Connect

```
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

```
for p in column:
    plt.figure()
    sns.boxplot(x=df[p])
```

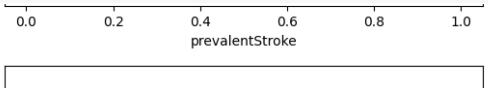


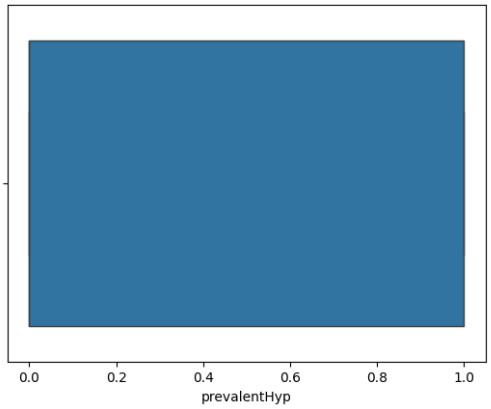


Commands + Code + Text Changes will not be saved Connect

'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
dtype='object')

for p in column:
plt.figure()
sns.boxplot(x=df[p])



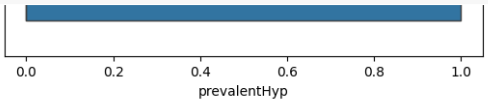


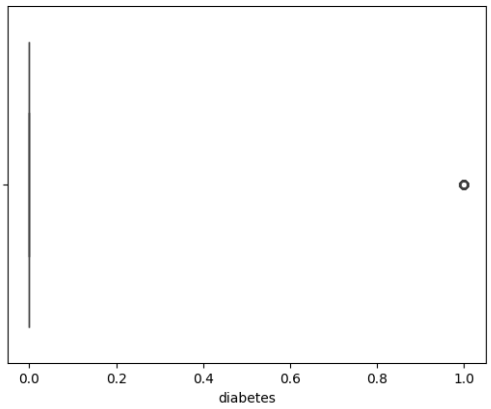
Activate Windows
Go to Settings to activate Windows.

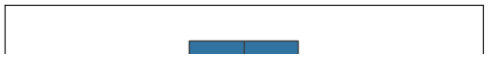
Commands + Code + Text Changes will not be saved Connect

```
      'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
     dtype='object')
```

```
for p in column:
    plt.figure()
    sns.boxplot(x=df[p])
```



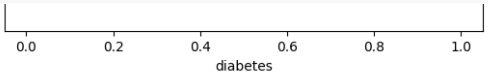


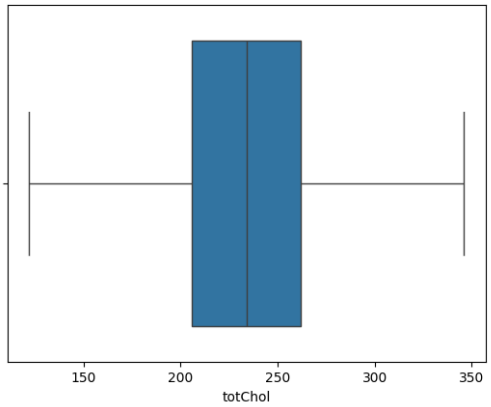


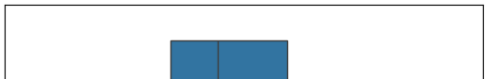
Commands + Code + Text Changes will not be saved Connect

'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
dtype='object')

for p in column:
plt.figure()
sns.boxplot(x=df[p])





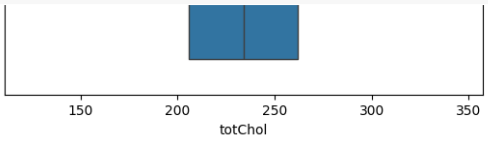


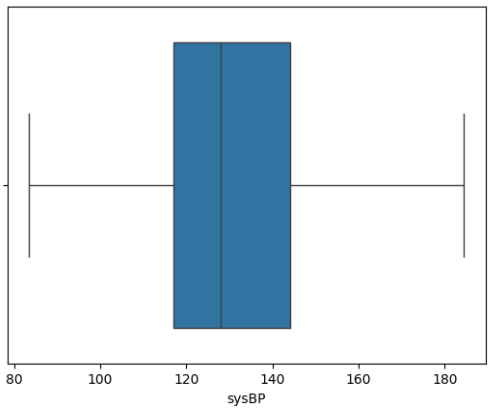
Activate Windows
Go to Settings to activate Windows.

Commands + Code + Text Changes will not be saved Connect

```
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

```
for p in column:
    plt.figure()
    sns.boxplot(x=df[p])
```



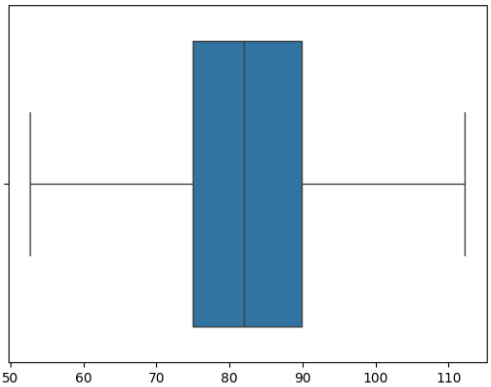


Commands + Code + Text Changes will not be saved Connect

'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
dtype='object')

for p in column:
plt.figure()
sns.boxplot(x=df[p])

80
100
120
140
160
180
sysBP

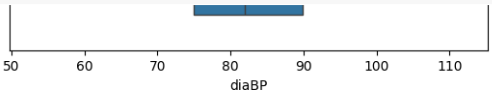


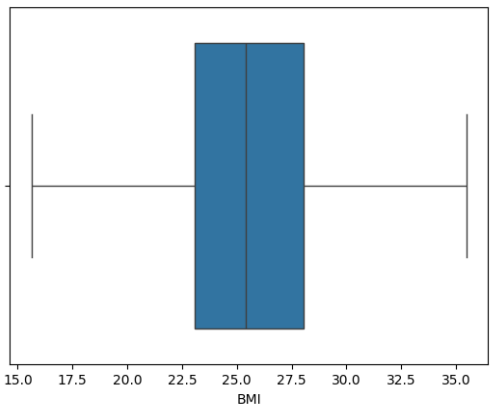
Activate Windows
Go to Settings to activate Windows.

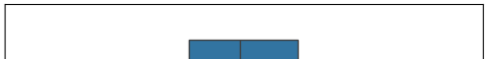
Commands + Code + Text Changes will not be saved Connect

```
      'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
     dtype='object')
```

```
for p in column:
    plt.figure()
    sns.boxplot(x=df[p])
```



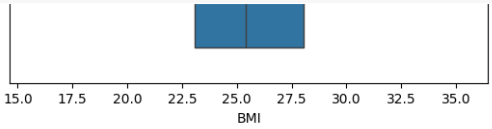


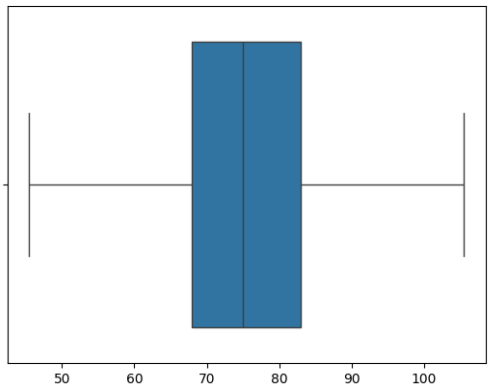


Commands + Code + Text Changes will not be saved Connect

'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
dtype='object')

for p in column:
plt.figure()
sns.boxplot(x=df[p])

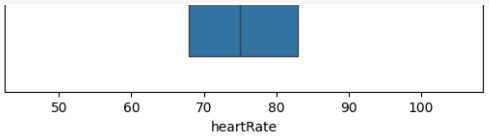


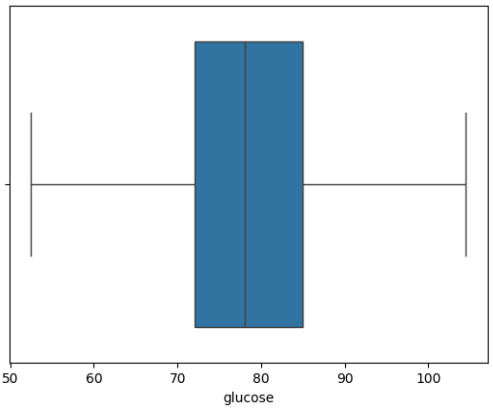


Commands + Code + Text Changes will not be saved Connect

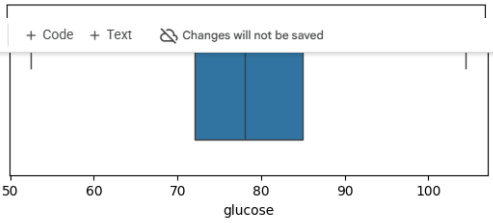
```
      'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
     dtype='object')
```

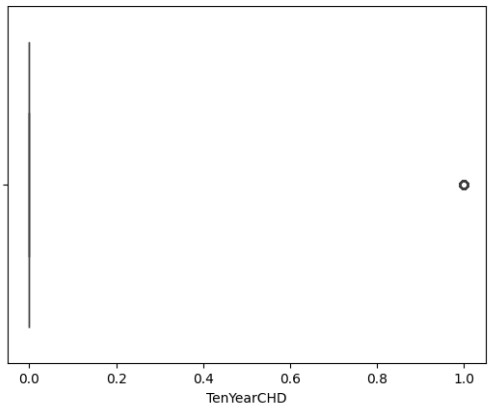
```
for p in column:
    plt.figure()
    sns.boxplot(x=df[p])
```





Commands + Code + Text Changes will not be saved Connect





Activate Windows
Go to Settings to activate Windows.

Now no outliers in dataset.

TenYearCHD

Now no outliers in dataset.

correlation in dataset

+ Code + Text

```
[ ] correlation= df.drop(['BPMeds','prevalentStroke','diabetes','TenYearCHD'],axis=1)
    corr=correlation.corr
    corr()
```

	male	age	education	currentSmoker	cigsPerDay	prevalentHyp	totChol	sysBP	diaBP	BMI	heartRate	glucose
male	1.000000	-0.028979	0.017205	0.197596	0.315879	0.005313	-0.071200	-0.030095	0.063325	0.102312	-0.116487	-0.001098
age	-0.028979	1.000000	-0.163547	-0.213748	-0.193645	0.307194	0.269226	0.397920	0.206682	0.137018	-0.014283	0.120069
education	0.017205	-0.163547	1.000000	0.018273	0.008041	-0.081021	-0.022037	-0.129073	-0.060799	-0.132060	-0.052507	-0.026354
currentSmoker	0.197596	-0.213748	0.018273	1.000000	0.765436	-0.103260	-0.049416	-0.130087	-0.107686	-0.170351	0.066365	-0.066862
cigsPerDay	0.315879	-0.193645	0.008041	0.765436	1.000000	-0.066566	-0.027134	-0.088021	-0.055622	-0.091968	0.077568	-0.080655
prevalentHyp	0.005313	0.307194	-0.081021	-0.103260	-0.066566	1.000000	0.160452	0.713008	0.622958	0.296577	0.147003	0.084638
totChol	-0.071200	0.269226	-0.022037	-0.049416	-0.027134	0.160452	1.000000	0.213093	0.174874	0.130529	0.088335	0.037241
sysBP	-0.030095	0.397920	-0.129073	-0.130087	-0.088021	0.713008	0.213093	1.000000	0.782594	0.319020	0.181762	0.119348
diaBP	0.063325	0.206682	-0.060799	-0.107686	-0.055622	0.622958	0.174874	0.782594	1.000000	0.370262	0.182718	0.051375
BMI	0.102312	0.137018	-0.132060	-0.170351	-0.091968	0.296577	0.130529	0.319020	0.370262	1.000000	0.060968	0.084017
heartRate	-0.116487	-0.014283	-0.052507	0.066365	0.077568	0.147003	0.088335	0.181762	0.182718	0.060968	1.000000	0.100291
glucose	-0.001098	0.120069	-0.026354	-0.066862	-0.080655	0.084638	0.037241	0.119348	0.051375	0.084017	0.100291	1.000000

Commands + Code + Text Changes will not be saved Connect

diaBP	0.063325	0.206682	-0.060799	-0.107686	-0.055622	0.622958	0.174874	0.782594	1.000000	0.370262	0.182718	0.051375
BMI	0.102312	0.137018	-0.132060	-0.170351	-0.091968	0.296577	0.130529	0.319020	0.370262	1.000000	0.060968	0.084017
heartRate	-0.116487	-0.014283	-0.052507	0.066365	0.077568	0.147003	0.088335	0.181762	0.182718	0.060968	1.000000	0.100291
glucose	-0.001098	0.120069	-0.026354	-0.066862	-0.080655	0.084638	0.037241	0.119348	0.051375	0.084017	0.100291	1.000000

good and positive correlation features

currentsmoker with cigsPerDay

prevalentHyp with sysBP

sysbp with diaBP

EDA - Exploratory data analysis

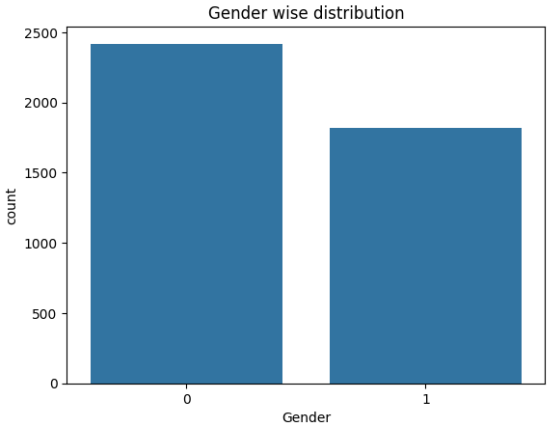
```
[ ] # gender distribution
    print("Total males are -",round(df['male'].value_counts(normalize=True)[1]*100),0)
    print("Total females are-",round(df['male'].value_counts(normalize=True)[0]*100),0)
```

```
Total males are - 43 0
Total females are- 57 0
```

Activate Windows
Go to Settings to activate Windows.

```
[ ] sns.countplot(x=df['male'])
    plt.title("Gender wise distribution")
    plt.xlabel("Gender")
    plt.show()
```

```
sns.countplot(x=df['male'])
plt.title("Gender wise distribution")
plt.xlabel("Gender")
plt.show()
```

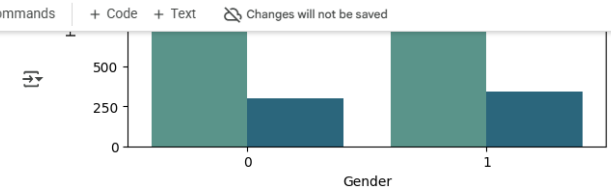



```
print("Total heart disease cases are - ",df['TenYearCHD'].value_counts()[1])
```

```
Total heart disease cases are -  644
```



```
sns.countplot(x=df['male'],hue=df['TenYearCHD'],palette='crest')
plt.ylabel("Heart Disease cases")
plt.xlabel("Gender")
plt.show()
```



301 No's female having 10 year risk of coronary heart disease CHD.

343 no's male having 10 year risk of coronary heart disease CHD

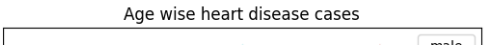
```
[ ] gen_chd= round(df[['male','TenYearCHD']].value_counts(normalize=True)*100,0)
    pd.DataFrame(gen_chd,columns=['percentage'])
```

	percentage

8% male and 7% female having 10 year risk of coronary heart disease CHD

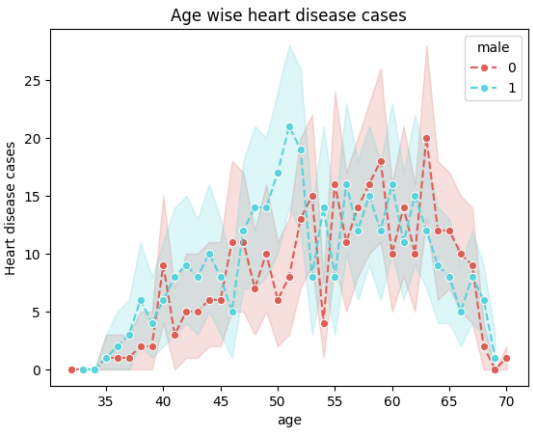
coronary heart disease in male is more than female.

```
[ ] sns.lineplot(data=df,x='age',y='TenYearCHD',estimator=sum,hue='male',linestyle='dashed',marker='o',palette='hls')
    plt.title("Age wise heart disease cases")
    plt.ylabel("Heart disease cases")
    plt.show()
```

Commands + Code + Text Changes will not be saved Connect

```
[ ] sns.lineplot(data=df,x='age',y='TenYearCHD',estimator=sum,hue='male',linestyle='dashed',marker='o',palette='hls')
    plt.title("Age wise heart disease cases")
    plt.ylabel("Heart disease cases")
    plt.show()
```

Male - Max heart disease cases in between 50 to 52 years old.

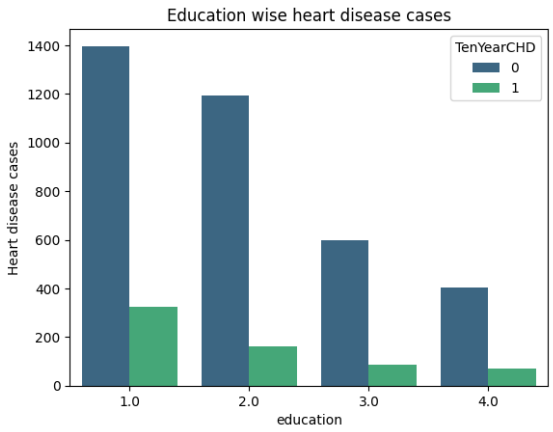
Female - Max heart disease cases in between 63 to 64 years old.

Commands + Code + Text Changes will not be saved Connect

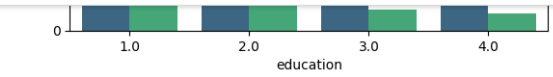
Male - Max heart disease cases in between 50 to 52 years old.

Female - Max heart disease cases in between 63 to 64 years old.

```
sns.countplot(data=df,x='education',hue='TenYearCHD',palette='viridis')
plt.title("Education wise heart disease cases")
plt.ylabel("Heart disease cases")
plt.show()
```



Commands + Code + Text Changes will not be saved Connect




```
[ ] round(df[['male','education']].value_counts(normalize=True)*100,0)
```

		proportion
male	education	
0	1.0	22.0
	2.0	19.0
1	1.0	18.0
	2.0	13.0
0	3.0	11.0
1	4.0	7.0
	3.0	5.0
0	4.0	5.0

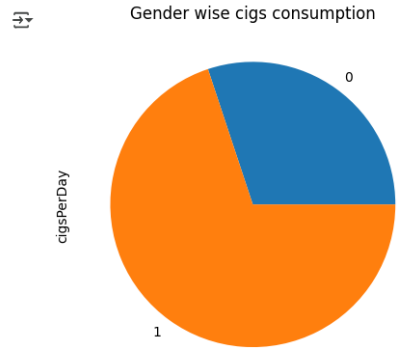
dtype: float64

Education 1.0 having maximum heart disease cases.

18% male are in education 1.0

Activate Windows
Go to Settings to activate Windows.

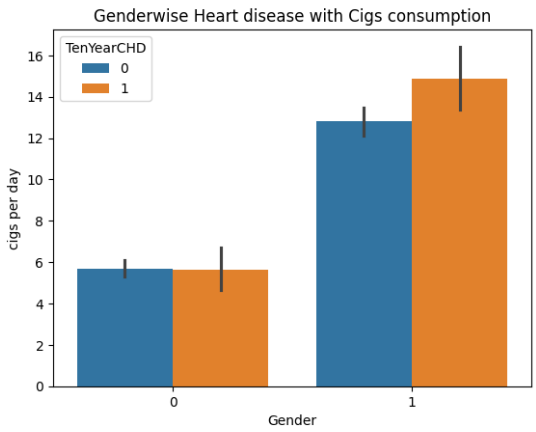
```
[ ] df.groupby('male')['cigsPerDay'].mean().plot(kind='pie')
    plt.title("Gender wise cigs consumption")
    plt.show()
```



Male consumed more cigarret than female.

average cigarret consumed by male per day is 13 no's.

```
[ ] sns.barplot(data=df,y='cigsPerDay',x='male',hue='TenYearCHD')
    plt.title("Genderwise Heart disease with Cigs consumption")
    plt.ylabel("cigs per day")
    plt.xlabel("Gender")
    plt.show()
```

male heart disease cases consumed more than 14 cigs per day.

female heart disease cases consumed 6 cigs per day.

even with less cigs consumption by female, only 1% difference in heart diseases count.

means cigs affecting more on female than male.

means cigs affecting more on female than male.

```
df.groupby('currentSmoker')['TenYearCHD'].sum()
```

currentSmoker	TenYearCHD
0	311
1	333

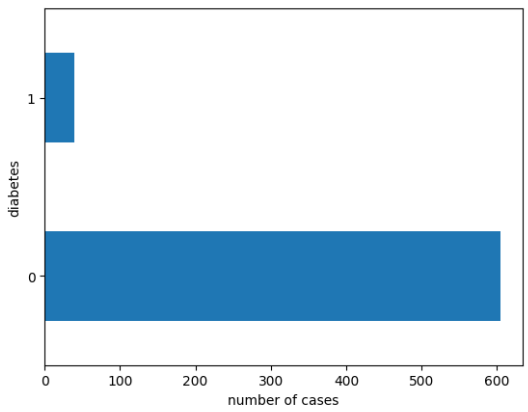
dtype: int64

no major diff in number of cases of current smoker and non current smoker.

```
df.groupby('diabetes')['TenYearCHD'].sum().plot(kind='barh')
plt.xlabel("number of cases")
plt.show()
```

no major diff in number of cases of current smoker and non current smoker.

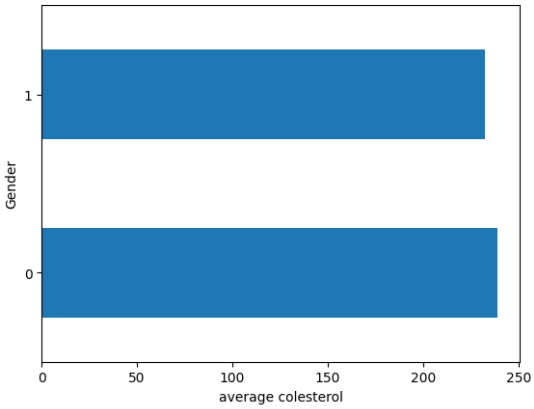
```
df.groupby('diabetes')['TenYearCHD'].sum().plot(kind='barh')
plt.xlabel("number of cases")
plt.show()
```



number of heart disease cases are more in non diabetes patient, so no any correlation between them.

Commands + Code + Text Changes will not be saved Connect

```
df.groupby('male')['totChol'].mean().plot(kind='barh')
plt.ylabel("Gender")
plt.xlabel("average colesterol")
plt.show()
```



no diff in avg colestrol of male and female.


```
[ ] print("average cholesterol of heart cases is - ",round(df.groupby('TenYearCHD')['totChol'].mean()[1],0))
```

```
average cholesterol of heart cases is -  244.0
```

Commands + Code + Text Changes will not be saved Connect

```
[ ] print("average body mass index of heart cases is - ",round(df.groupby('TenYearCHD')['BMI'].mean()[1],0))
```

```
average body mass index of heart cases is -  26.0
```

```
[ ] print("average glucose of heart cases is - ",round(df.groupby('TenYearCHD')['glucose'].mean()[1],0))
```

```
average glucose of heart cases is -  81.0
```

```
[ ] print("average heartRate of heart cases is - ",round(df.groupby('TenYearCHD')['heartRate'].mean()[1],0))
```

```
average heartRate of heart cases is -  76.0
```

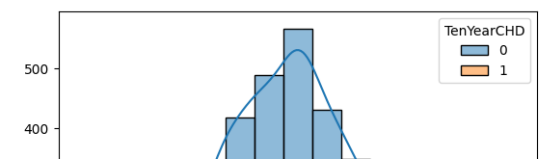
```
[ ] print("average sysBP of heart cases is - ",round(df.groupby('TenYearCHD')['sysBP'].mean()[1],0))
```

```
average sysBP of heart cases is -  142.0
```

```
[ ] print("average diaBP of heart cases is - ",round(df.groupby('TenYearCHD')['diaBP'].mean()[1],0))
```

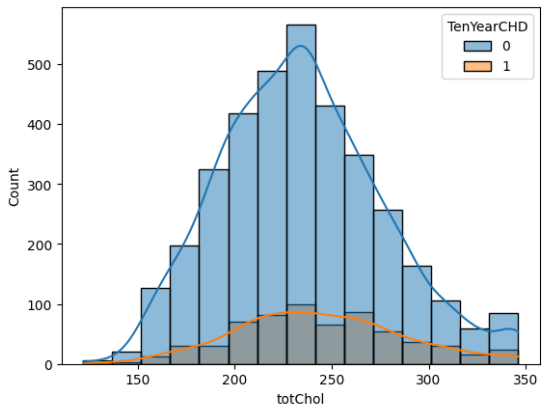
```
average diaBP of heart cases is -  87.0
```

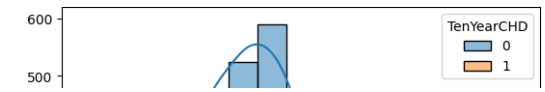
```
[ ] dist=df[['totChol', 'sysBP',
           'diaBP', 'BMI', 'heartRate', 'glucose']]
    for i in dist:
        plt.figure()
        sns.histplot(x=dist[i],hue=df['TenYearCHD'],kde=True,bins=15)
```

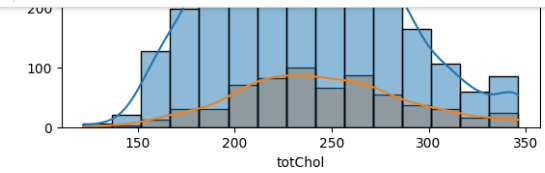


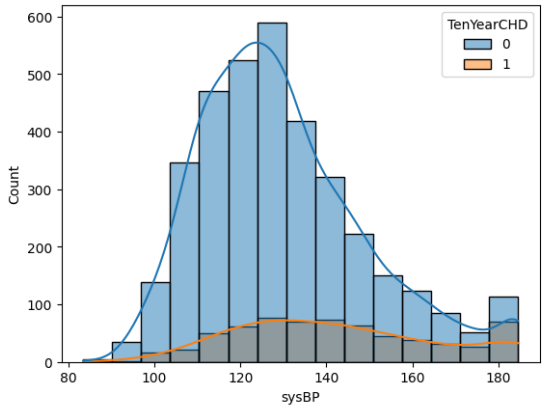
Activate Windows
Go to Settings to activate Windows.

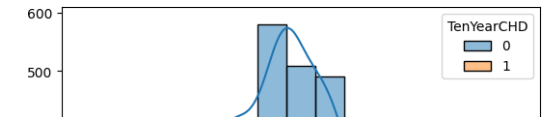
```
[ ] print("average diaBP of heart cases is - ",round(df.groupby('TenYearCHD')['diaBP'].mean()[1],0))

    average diaBP of heart cases is -  87.0

[ ] dist=df[['totChol', 'sysBP',
             'diaBP', 'BMI', 'heartRate', 'glucose']]
    for i in dist:
        plt.figure()
        sns.histplot(x=dist[i],hue=df['TenYearCHD'],kde=True,bins=15)
```

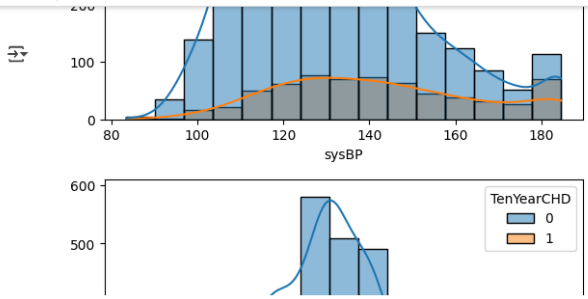












max heart disease cases fall under cholestrol range of 220 to 260.

Max heart disease cases fall under MBI range 23.5 to 27.5.

Max heart disease cases fall under sysBP range 125 to 145.

Max heart disease cases fall under diaBP range 80 to 90.

max heart disease cases fall under heart rate range 73 to 76.

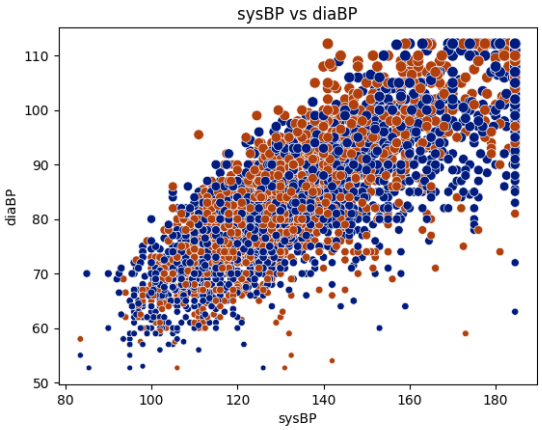
max heart disease cases fall under glucose range 78 to 81.


```
[ ] sns.scatterplot(data=df,x='sysBP',y='diaBP',size='diaBP',legend=None,hue='male',palette='dark')
    plt.title("sysBP vs diaBP")
    plt.show()
```

max heart disease cases fall under heart rate range 73 to 76.

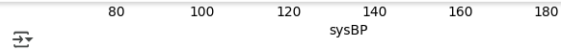
max heart disease cases fall under glucose range 78 to 81.

[ ] sns.scatterplot(data=df,x='sysBP',y='diaBP',size='diaBP',legend=None,hue='male',palette='dark')
plt.title("sysBP vs diaBP")
plt.show()



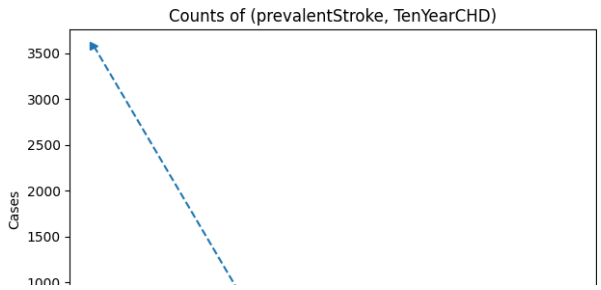
Activate Windows
Go to Settings to activate Windows.

sysBP and diaBP having positive relationship.



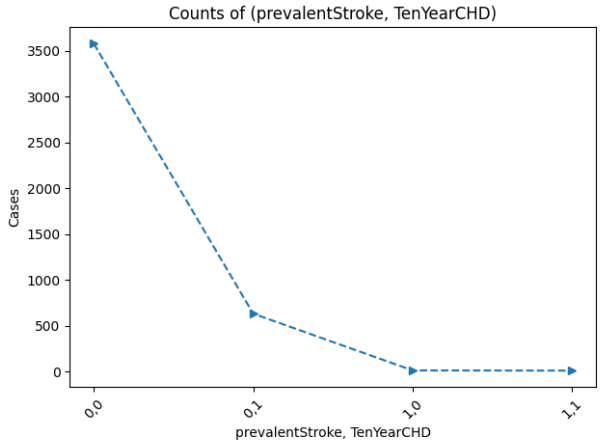
sysBP and diaBP having positive relationship.

```
[ ] # Step 1: Get value counts and reset index
    data = df[['prevalentStroke', 'TenYearCHD']].value_counts().reset_index(name='cases')

    # Step 2: Plot
    plt.plot(data['cases'], linestyle='dashed', marker='>')
    plt.xticks(ticks=range(len(data)),
               labels=[f"{a},{b}" for a, b in zip(data['prevalentStroke'], data['TenYearCHD'])],
               rotation=45)
    plt.xlabel("prevalentStroke, TenYearCHD")
    plt.ylabel("Cases")
    plt.title("Counts of (prevalentStroke, TenYearCHD)")
    plt.tight_layout()
    plt.show()
```



Commands + Code + Text Changes will not be saved Connect

```
[ ] plt.title("Counts of (prevalentStroke, TenYearCHD)")
    plt.tight_layout()
    plt.show()
```



Activate Windows
Go to Settings to activate Windows.

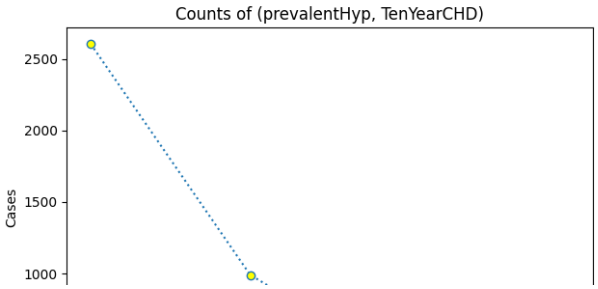
highest heart disease cases having no prevalent stroke.

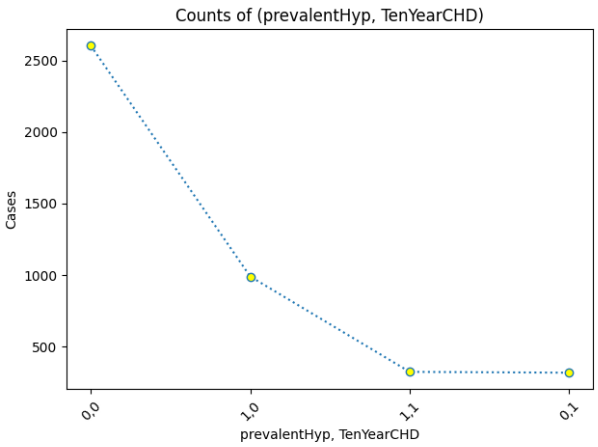
No any relation between them.

highest heart disease cases having no prevalent stroke.

No any relation between them.

```
[ ] # Step 1: Count combinations and reset index
    data = df[['prevalentHyp', 'TenYearCHD']].value_counts().reset_index(name='cases')

    # Step 2: Plot
    plt.plot(data['cases'], linestyle='dotted', marker='o', mfc='yellow')
    plt.xticks(ticks=range(len(data)),
               labels=[f"{a},{b}" for a, b in zip(data['prevalentHyp'], data['TenYearCHD'])],
               rotation=45)
    plt.xlabel("prevalentHyp, TenYearCHD")
    plt.ylabel("Cases")
    plt.title("Counts of (prevalentHyp, TenYearCHD)")
    plt.tight_layout()
    plt.show()
```





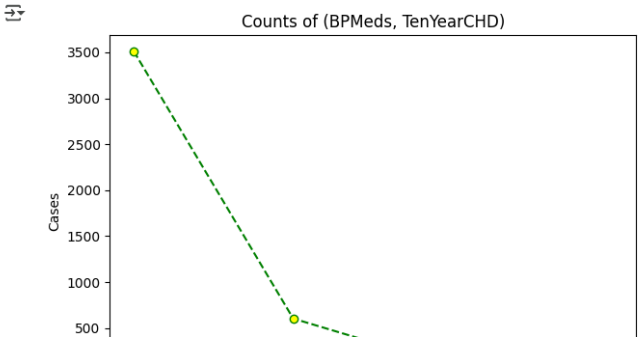
No any relation of prevalentHyp and heart disease.

```
[ ] # Step 1: Get counts and reset index
    data = df[['BPMeds', 'TenYearCHD']].value_counts().reset_index(name='cases')

    # Step 2: Plot
    plt.plot(data['cases'], linestyle='dashed', marker='o', color='green', mfc='yellow')
    plt.xticks(ticks=range(len(data)),
```

Commands + Code + Text Changes will not be saved Connect

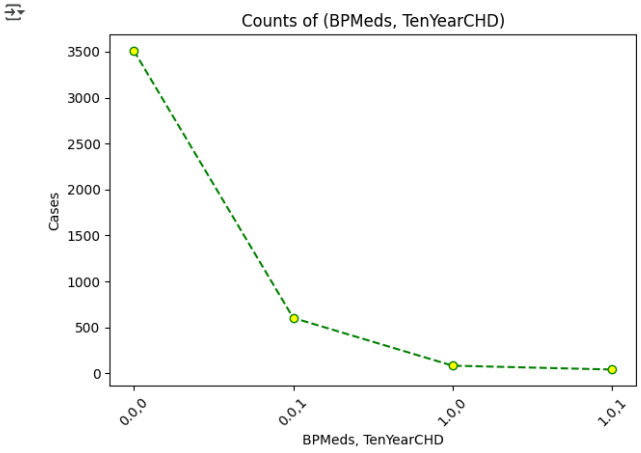
No any relation of prevalentHyp and heart disease.

```
[ ] # Step 1: Get counts and reset index
    data = df[['BPMeds', 'TenYearCHD']].value_counts().reset_index(name='cases')

    # Step 2: Plot
    plt.plot(data['cases'], linestyle='dashed', marker='o', color='green', mfc='yellow')
    plt.xticks(ticks=range(len(data)),
               labels=[f"{a},{b}" for a, b in zip(data['BPMeds'], data['TenYearCHD'])],
               rotation=45)
    plt.xlabel("BPMeds, TenYearCHD")
    plt.ylabel("Cases")
    plt.title("Counts of (BPMeds, TenYearCHD)")
    plt.tight_layout()
    plt.show()
```



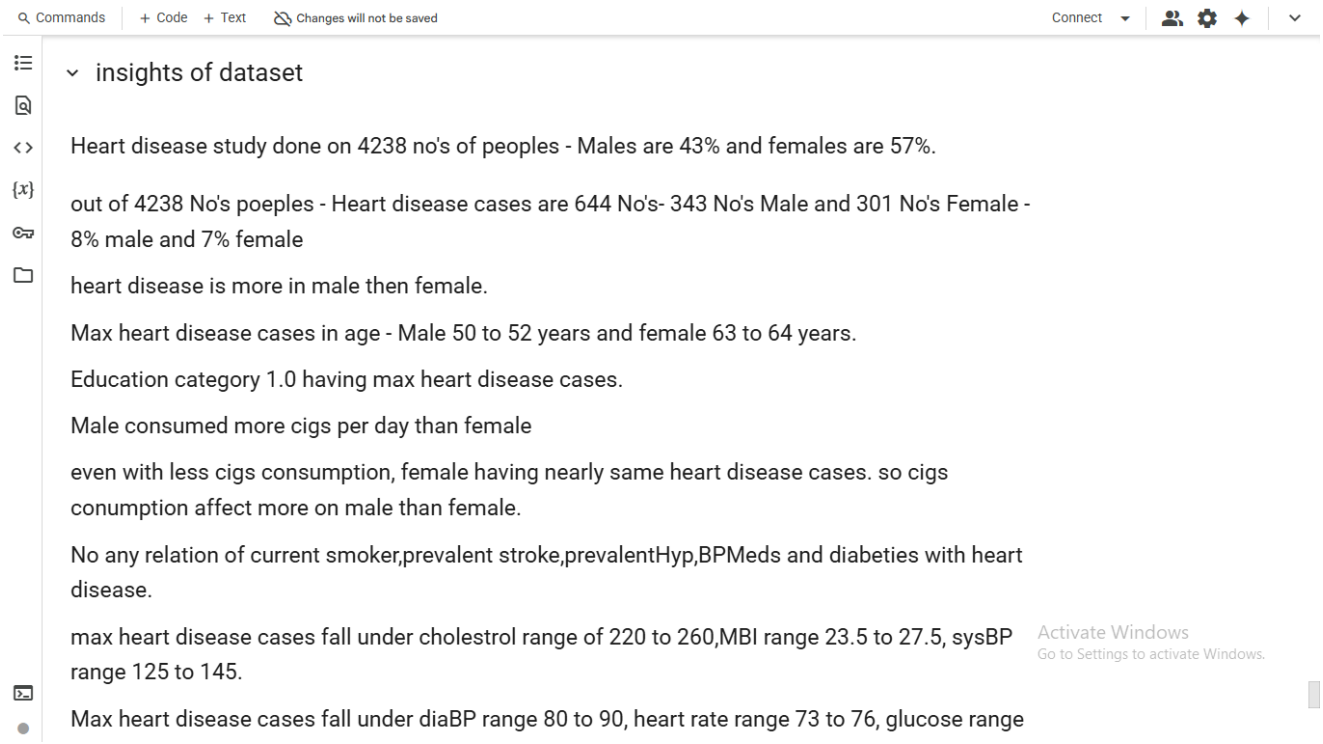
Commands + Code + Text Changes will not be saved Connect

```
[ ] plt.title("Counts of (BPMeds, TenYearCHD)")
    plt.tight_layout()
    plt.show()
```



No any relation in BPMeds and heart disease.

insights of dataset



5. CONCLUSION

This study analyzed the Framingham Heart Study dataset to identify risk factors associated with coronary heart disease (CHD) within 10 years. The analysis revealed the following key insights:

- **Demographics:** The dataset included 4238 individuals, with males comprising 43% and females 57%. Heart disease cases were more prevalent in males (8%) compared to females (7%).
- **Age:** The risk of CHD increased with age, with peak prevalence observed in males between 50-52 years and in females between 63-64 years.

- **Risk Factors:**
    - Education level 1.0 was associated with a higher risk of CHD.
    - Cigarette consumption was higher in males, and both male and female smokers showed an increased risk of CHD. However, the impact of smoking appeared to be more pronounced in females despite lower consumption.
    - Current smoking status, prevalent stroke, prevalent hypertension, blood pressure medication usage, and diabetes did not show a strong correlation with CHD risk.
    - Elevated levels of cholesterol (220-260), BMI (23.5-27.5), systolic blood pressure (125-145), diastolic blood pressure (80-90), heart rate (73-76), and glucose (78-81) were linked to increased CHD risk.
- **Machine Learning Model:** A logistic regression model was developed to predict 10-year CHD risk based on selected features. The model achieved an accuracy of 85%.

Overall, this analysis highlighted several important risk factors for CHD, including age, gender, education, cigarette consumption, and various physiological measures. The developed logistic regression model provides a promising tool for identifying individuals at risk of developing CHD within the next 10 years.

Further Research: Future research could explore the impact of other potential risk factors not included in this study. Additionally, model performance could be further improved by exploring alternative machine learning algorithms and feature engineering techniques

THANK YOU