```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


df=pd.read_csv('/content/CVD_cleaned[1].csv')
df
```

l_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diab
Poor	Within the past 2 years	No	No	No	No	No	
/ery Good	Within the past year	No	Yes	No	No	No	
/ery Good	Within the past year	Yes	No	No	No	No	
Poor	Within the past year	Yes	Yes	No	No	No	
Good	Within the past year	No	No	No	No	No	
...	...	...	...	...	...	...	
/ery Good	Within the past year	Yes	No	No	No	No	
Fair	Within the past 5 years	Yes	No	No	No	No	
/ery Good	5 or more years ago	Yes	No	No	No	Yes	Yes fe told d pregn
/ery Good	Within the past year	Yes	No	No	No	No	
Excellent	Within the past year	Yes	No	No	No	No	

columns

```
df.columns
```

```
Index(['General_Health', 'Checkup', 'Exercise', 'Heart_Disease', 'Skin_Cancer',
       'Other_Cancer', 'Depression', 'Diabetes', 'Arthritis', 'Sex',
       'Age_Category', 'Height_(cm)', 'Weight_(kg)', 'BMI', 'Smoking_History',
       'Alcohol_Consumption', 'Fruit_Consumption',
       'Green_Vegetables_Consumption', 'FriedPotato_Consumption'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 308854 entries, 0 to 308853
Data columns (total 19 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   General_Health                308854 non-null  object 
 1   Checkup                       308854 non-null  object 
 2   Exercise                      308854 non-null  object 
 3   Heart_Disease                 308854 non-null  object 
 4   Skin_Cancer                   308854 non-null  object 
 5   Other_Cancer                  308854 non-null  object 
 6   Depression                    308854 non-null  object 
 7   Diabetes                      308854 non-null  object 
 8   Arthritis                     308854 non-null  object 
 9   Sex                           308854 non-null  object 
 10  Age_Category                  308854 non-null  object 
 11  Height_(cm)                   308854 non-null  float64
 12  Weight_(kg)                   308854 non-null  float64
 13  BMI                           308854 non-null  float64
 14  Smoking_History               308854 non-null  object 
 15  Alcohol_Consumption           308854 non-null  float64
 16  Fruit_Consumption             308854 non-null  float64
 17  Green_Vegetables_Consumption  308854 non-null  float64
 18  FriedPotato_Consumption       308854 non-null  float64
dtypes: float64(7), object(12)
memory usage: 44.8+ MB
```

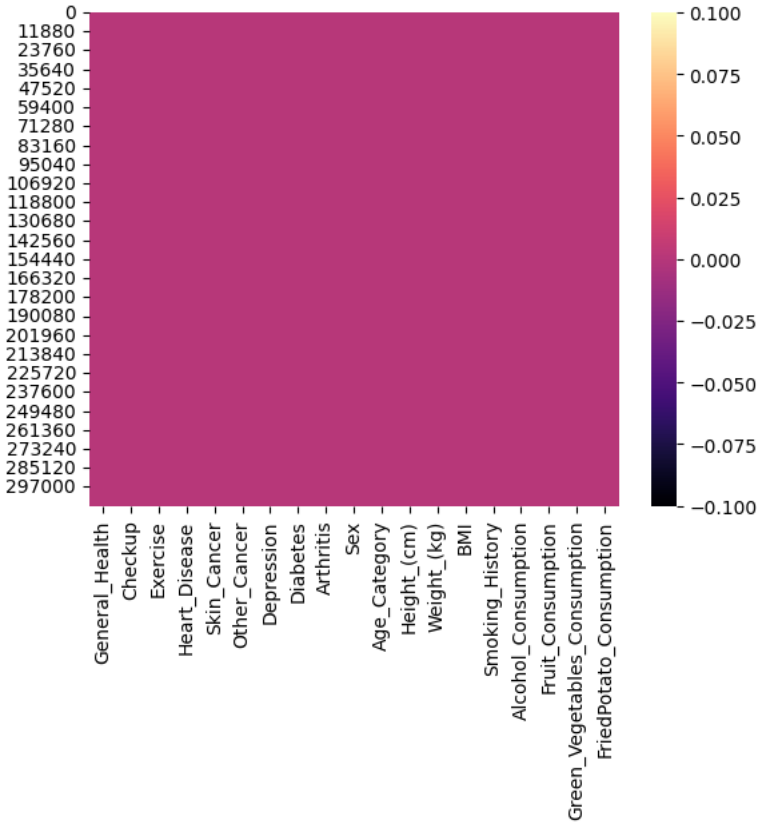
```
df.isna().sum()
```

```
General_Health                  0
Checkup                         0
Exercise                        0
Heart_Disease                   0
Skin_Cancer                     0
Other_Cancer                    0
Depression                      0
Diabetes                        0
Arthritis                       0
Sex                             0
Age_Category                    0
Height_(cm)                     0
Weight_(kg)                     0
BMI                             0
Smoking_History                 0
Alcohol_Consumption             0
Fruit_Consumption               0
Green_Vegetables_Consumption    0
```

```
FriedPotato_Consumption         0
dtype: int64
```

```
sns.heatmap(df.isnull(), cmap='magma')
```

```
<Axes: >
```



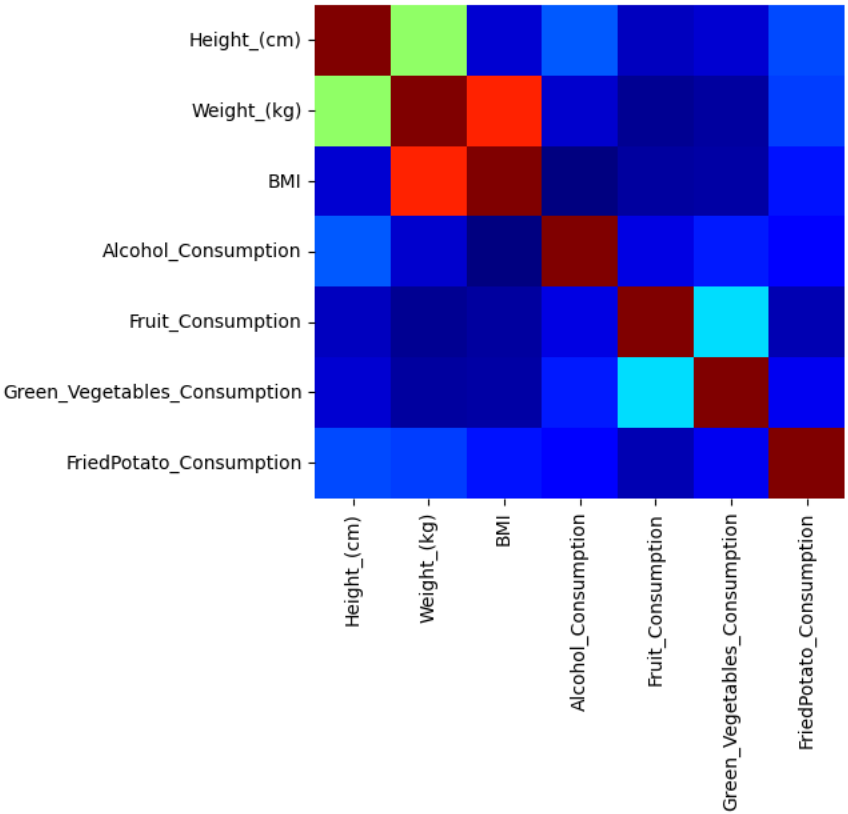

```
cor=df.corr()
cor
```

```
<ipython-input-11-7a446f931109>:1: FutureWarning: The default value of numeric_only ir
  cor=df.corr()
```

	Height_(cm)	Weight_(kg)	BMI	Alcohol_Consumption
Height_(cm)	1.000000	0.472186	-0.027408	0.128835

```
sns.heatmap(cor, cmap='jet')
```

```
<Axes: >
```




```
#value counts
col=df.columns
for col in df.columns:
  print('Value counts of', col, 'is', '\n', df[col].value_counts())
```

```
Value counts of General_Health is
 Very Good    110395
Good          95364
Excellent     55954
Fair          35810
Poor          11331
Name: General_Health, dtype: int64
Value counts of Checkup is
```

```
 Within the past year       239371
Within the past 2 years     37213
Within the past 5 years     17442
5 or more years ago         13421
Never                        1407
Name: Checkup, dtype: int64
Value counts of Exercise is
 Yes    239381
No      69473
Name: Exercise, dtype: int64
Value counts of Heart_Disease is
 No     283883
Yes     24971
Name: Heart_Disease, dtype: int64
Value counts of Skin_Cancer is
 No     278860
Yes     29994
Name: Skin_Cancer, dtype: int64
Value counts of Other_Cancer is
 No     278976
Yes     29878
Name: Other_Cancer, dtype: int64
Value counts of Depression is
 No     246953
Yes     61901
Name: Depression, dtype: int64
Value counts of Diabetes is
 No                                          259141
Yes                                         40171
No, pre-diabetes or borderline diabetes      6896
Yes, but female told only during pregnancy   2646
Name: Diabetes, dtype: int64
Value counts of Arthritis is
 No     207783
Yes    101071
Name: Arthritis, dtype: int64
Value counts of Sex is
 Female    160196
Male      148658
Name: Sex, dtype: int64
Value counts of Age_Category is
 65-69    33434
60-64    32418
70-74    31103
55-59    28054
50-54    25097
80+      22271
40-44    21595
45-49    20968
75-79    20705
```

```
#unique values in each column]
col=df.columns
for col in (df.columns):
    print ("Unique columns of", col, "\n", df[col].unique())
    print ("--------------")
```

```
 Unique columns of General_Health
  ['Poor' 'Very Good' 'Good' 'Fair' 'Excellent']
 --------------
 Unique columns of Checkup
```

```
 ['Within the past 2 years' 'Within the past year' '5 or more years ago'
 'Within the past 5 years' 'Never']
--------------
Unique columns of Exercise
 ['No' 'Yes']
--------------
Unique columns of Heart_Disease
 ['No' 'Yes']
--------------
Unique columns of Skin_Cancer
 ['No' 'Yes']
--------------
Unique columns of Other_Cancer
 ['No' 'Yes']
--------------
Unique columns of Depression
 ['No' 'Yes']
--------------
Unique columns of Diabetes
 ['No' 'Yes' 'No, pre-diabetes or borderline diabetes'
 'Yes, but female told only during pregnancy']
--------------
Unique columns of Arthritis
 ['Yes' 'No']
--------------
Unique columns of Sex
 ['Female' 'Male']
--------------
Unique columns of Age_Category
 ['70-74' '60-64' '75-79' '80+' '65-69' '50-54' '45-49' '18-24' '30-34'
 '55-59' '35-39' '40-44' '25-29']
--------------
Unique columns of Height_(cm)
 [150. 165. 163. 180. 191. 183. 175. 160. 168. 178. 152. 157. 188. 185.
 170. 173. 155. 193. 196. 206. 198. 140. 135. 145. 147. 142. 201. 218.
 124. 203. 137. 122. 216. 224. 229. 151. 177. 164. 162. 156. 153. 169.
 167. 172. 106. 190. 143. 171. 154. 176. 200. 146. 148. 158. 159. 187.
 104. 120. 107. 211. 226. 182. 213.  97. 184. 125. 127. 234. 130. 119.
 132. 105. 166. 181. 186.  91. 174. 208. 149.  96. 197. 161.  94. 103.
 221. 134. 144. 189. 100. 179. 117.  99. 102. 110. 241. 115. 205. 195.
 108.]
--------------
Unique columns of Weight_(kg)
 [ 32.66  77.11  88.45  93.44 154.22  69.85 108.86  72.57  91.63  74.84
  73.48  83.91 113.4   52.16 116.12  99.79  81.65 104.33  79.38  55.79
 124.74  81.19  70.31 112.49 147.42  84.82 102.06  64.41  60.78  61.23
  88.    90.72  49.9   85.28 120.2   69.4   62.14  65.77  89.81  66.68
  86.18  72.12  87.54  62.6   75.75  88.9   92.08  56.7   68.04  79.83
  63.5   58.97 114.76  45.36  73.94  54.43 125.19  77.56  96.16  95.25
 115.67  82.55 136.08  78.93  70.76  95.71  53.52  87.09  55.34  83.01
 123.38  98.88  73.03  76.66  97.52  71.67  83.46 122.47  58.06  74.39
  67.13  82.1   47.63  99.34  85.73 108.41  91.17  57.61  63.05  45.81
  94.35  44.45 117.93 107.5  127.01 106.59 107.95  89.36  92.99  53.07
  78.02 131.09  97.98  84.37 111.13  50.8   57.15  64.86  80.29  76.2
```

```
df['General_Health'].unique()
```

```
array(['Poor', 'Very Good', 'Good', 'Fair', 'Excellent'], dtype=object)
```

```
#label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

for col in df.columns:
  if df[col].dtypes == 'object':
    df[col]=le.fit_transform(df[col])

df
```

ɔn	Diabetes	Arthritis	Sex	Age_Category	Height_(cm)	Weight_(kg)	BMI	Smoking_Hist
0	0	1	0	10	150.0	32.66	14.54	
0	2	0	0	10	165.0	77.11	28.29	
0	2	0	0	8	163.0	88.45	33.47	
0	2	0	1	11	180.0	93.44	28.73	
0	0	0	1	12	191.0	88.45	24.37	
...	...	...	...	...	...	...	...	
0	0	0	1	1	168.0	81.65	29.05	
0	2	0	1	9	180.0	69.85	21.48	
1	3	0	0	2	157.0	61.23	24.69	
0	0	0	1	9	183.0	79.38	23.73	
0	0	0	0	5	160.0	81.19	31.71	

```
df.dtypes
```

```
General_Health                  int64
Checkup                         int64
Exercise                        int64
Heart_Disease                   int64
Skin_Cancer                     int64
Other_Cancer                    int64
Depression                      int64
Diabetes                        int64
Arthritis                       int64
Sex                             int64
Age_Category                    int64
Height_(cm)                   float64
Weight_(kg)                   float64
BMI                           float64
Smoking_History                 int64
Alcohol_Consumption           float64
Fruit_Consumption             float64
Green_Vegetables_Consumption  float64
FriedPotato_Consumption       float64
dtype: object
```

```
x=df.drop(['Heart_Disease'], axis=1)
x
```

	General_Health	Checkup	Exercise	Skin_Cancer	Other_Cancer	Depression	Diab
0	3	2	0	0	0	0	
1	4	4	0	0	0	0	
2	4	4	1	0	0	0	
3	3	4	1	0	0	0	
4	2	4	0	0	0	0	
...	...	...	...	...	...	...	
308849	4	4	1	0	0	0	
308850	1	3	1	0	0	0	
308851	4	0	1	0	0	1	
308852	4	4	1	0	0	0	
308853	0	4	1	0	0	0	

308854 rows × 18 columns

```
y=df['Heart_Disease']
y
```

```
0         0
1         1
2         0
3         1
4         0
         ..
308849    0
308850    0
308851    0
308852    0
308853    0
Name: Heart_Disease, Length: 308854, dtype: int64
```

Visualization

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.1, random_state=42)
x_train.shape
```

```
(277968, 18)
```

```
x_test.shape
```

```
(30886, 18)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

```
#ANN actual training
!pip install -q keras
```

```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

```
model = Sequential()
model.add(Dense(64, input_dim=x_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
#Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

#Train the model
model.fit(x_train, y_train, epochs=50, batch_size=32, validation_split=0.2)

#Evaluate the model on the test set
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Test accuracy: {accuracy:.4f}")
```

```
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2159 - accuracy
Epoch 33/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2157 - accuracy
Epoch 34/50
6950/6950 [==============================] - 17s 2ms/step - loss: 0.2157 - accuracy
Epoch 35/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2156 - accuracy
Epoch 36/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2154 - accuracy
Epoch 37/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2153 - accuracy
Epoch 38/50
6950/6950 [==============================] - 17s 2ms/step - loss: 0.2152 - accuracy
Epoch 39/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2150 - accuracy
Epoch 40/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2151 - accuracy
Epoch 41/50
6950/6950 [==============================] - 19s 3ms/step - loss: 0.2148 - accuracy
Epoch 42/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2148 - accuracy
Epoch 43/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2147 - accuracy
Epoch 44/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2145 - accuracy
Epoch 45/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2145 - accuracy
Epoch 46/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2145 - accuracy
Epoch 47/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2142 - accuracy
Epoch 48/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2141 - accuracy
Epoch 49/50
6950/6950 [==============================] - 16s 2ms/step - loss: 0.2142 - accuracy
```

```
#next to find prediction and confusion matrix
y_pred=model.predict(x_test)
y_pred=(y_pred>0.5)
```

```
966/966 [==============================] - 2s 2ms/step
```

```
y_pred
```

```
array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]])
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
```

```
cm
```

```
array([[28273,   130],
       [ 2380,   103]])
```

```
#Convert output to pandas Series
y_pred=y_pred.ravel()
#Convert predicted probabilities to binary predictions (0 or 1)
y_pred_binary=np.round(y_pred).astype(int)
import pandas as pd
results=pd.DataFrame()
results["Actual value"]=y_test
results["Predicted value"]=y_pred_binary
results["Difference"]=y_test-y_pred_binary
results["Prediction Correct"] = np.where(results["Actual value"] == results["Predicted valu
results.sort_index()
```

1 to 25 of 20000 entries Filter

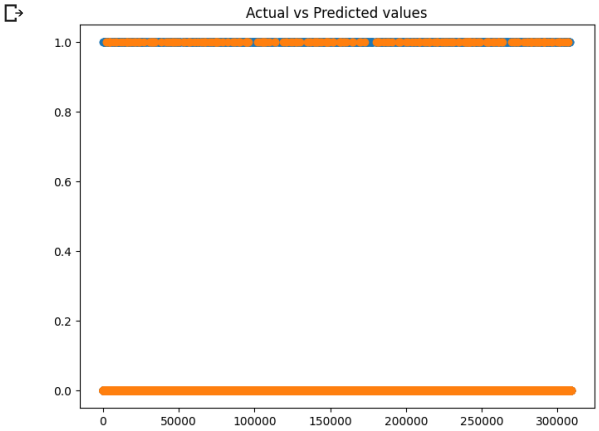
index	Actual value	Predicted value	Difference	Prediction Cor
0	0	0	0	Good
2	0	0	0	Good
16	0	0	0	Good
22	0	0	0	Good
24	0	0	0	Good
26	0	0	0	Good
30	0	0	0	Good
41	0	0	0	Good
49	0	0	0	Good
53	0	0	0	Good
85	0	0	0	Good
87	0	0	0	Good
88	1	0	1	Bad
106	0	0	0	Good
117	0	0	0	Good
118	0	0	0	Good
119	0	0	0	Good
123	0	0	0	Good
128	0	0	0	Good
139	0	0	0	Good
142	0	0	0	Good
172	0	0	0	Good
183	0	0	0	Good
197	0	0	0	Good
201	0	0	0	Good

Show 25 per page

1 2 10 100 700 790 800

```
plt.figure(figsize=(8,6))
plt.scatter(results.index, results["Actual value"], label="Actual")
plt.scatter(results.index, results["Predicted value"], label="Predicted")
```

```
plt.xlabel=("Index")
plt.ylabel=("Values")
plt.title("Actual vs Predicted values")
plt.show()
```



Hence, we got good accuracy