```
import pandas as pd

df = pd.read_csv(r'C:\Users\LENOVO\Downloads\archive (2)\
alzheimers_disease_data.csv')
print(df)
```

```
      PatientID  Age  Gender  Ethnicity  EducationLevel        BMI
Smoking  \
0          4751   73       0          0               2  22.927749
0
1          4752   89       0          0               0  26.827681
0
2          4753   73       0          3               1  17.795882
0
3          4754   74       1          0               1  33.800817
1
4          4755   89       0          0               0  20.716974
0
...         ...  ...     ...        ...             ...        ...
...
2144       6895   61       0          0               1  39.121757
0
2145       6896   75       0          0               2  17.857903
0
2146       6897   77       0          0               1  15.476479
0
2147       6898   78       1          3               1  15.299911
0
2148       6899   72       0          0               2  33.289738
0

      AlcoholConsumption  PhysicalActivity  DietQuality  ...  \
0              13.297218          6.327112     1.347214  ...
1               4.542524          7.619885     0.518767  ...
2              19.555085          7.844988     1.826335  ...
3              12.209266          8.428001     7.435604  ...
4              18.454356          6.310461     0.795498  ...
...                  ...               ...          ...  ...
2144            1.561126          4.049964     6.555306  ...
2145           18.767261          1.360667     2.904662  ...
2146            4.594670          9.886002     8.120025  ...
2147            8.674505          6.354282     1.263427  ...
2148            7.890703          6.570993     7.941404  ...

      MemoryComplaints  BehavioralProblems       ADL  Confusion  \
0                    0                   0  1.725883          0
1                    0                   0  2.592424          0
2                    0                   0  7.119548          0
3                    0                   1  6.481226          0
4                    0                   0  0.014691          0
```

```
...                 ...                 ...       ...         ...
2144                  0                   0  4.492838           1
2145                  0                   1  9.204952           0
2146                  0                   0  5.036334           0
2147                  0                   0  3.785399           0
2148                  0                   1  8.327563           0

      Disorientation  PersonalityChanges  DifficultyCompletingTasks  \
0                  0                   0                          1
1                  0                   0                          0
2                  1                   0                          1
3                  0                   0                          0
4                  0                   1                          1
...              ...                 ...                        ...
2144               0                   0                          0
2145               0                   0                          0
2146               0                   0                          0
2147               0                   0                          0
2148               1                   0                          0

      Forgetfulness  Diagnosis  DoctorInCharge
0                 0          0       XXXConfid
1                 1          0       XXXConfid
2                 0          0       XXXConfid
3                 0          0       XXXConfid
4                 0          0       XXXConfid
...             ...        ...             ...
2144              0          1       XXXConfid
2145              0          1       XXXConfid
2146              0          1       XXXConfid
2147              1          1       XXXConfid
2148              1          0       XXXConfid

[2149 rows x 35 columns]

df = df.drop_duplicates()
df = df.reset_index(drop=True)
print(df)

      PatientID  Age  Gender  Ethnicity  EducationLevel        BMI
Smoking  \
0          4751   73       0          0               2  22.927749
0
1          4752   89       0          0               0  26.827681
0
2          4753   73       0          3               1  17.795882
0
3          4754   74       1          0               1  33.800817
1
4          4755   89       0          0               0  20.716974
```

```
0
...         ...  ...     ...        ...              ...        ...
...
2144       6895   61       0          0                1  39.121757
0
2145       6896   75       0          0                2  17.857903
0
2146       6897   77       0          0                1  15.476479
0
2147       6898   78       1          3                1  15.299911
0
2148       6899   72       0          0                2  33.289738
0

      AlcoholConsumption  PhysicalActivity  DietQuality  ...  \
0              13.297218          6.327112     1.347214  ...
1               4.542524          7.619885     0.518767  ...
2              19.555085          7.844988     1.826335  ...
3              12.209266          8.428001     7.435604  ...
4              18.454356          6.310461     0.795498  ...
...                  ...               ...          ...  ...
2144            1.561126          4.049964     6.555306  ...
2145           18.767261          1.360667     2.904662  ...
2146            4.594670          9.886002     8.120025  ...
2147            8.674505          6.354282     1.263427  ...
2148            7.890703          6.570993     7.941404  ...

      MemoryComplaints  BehavioralProblems       ADL  Confusion  \
0                    0                   0  1.725883          0
1                    0                   0  2.592424          0
2                    0                   0  7.119548          0
3                    0                   1  6.481226          0
4                    0                   0  0.014691          0
...                ...                 ...       ...        ...
2144                 0                   0  4.492838          1
2145                 0                   1  9.204952          0
2146                 0                   0  5.036334          0
2147                 0                   0  3.785399          0
2148                 0                   1  8.327563          0

      Disorientation  PersonalityChanges  DifficultyCompletingTasks  \
0                  0                   0                          1
1                  0                   0                          0
2                  1                   0                          1
3                  0                   0                          0
4                  0                   1                          1
...              ...                 ...                        ...
2144               0                   0                          0
2145               0                   0                          0
2146               0                   0                          0
```

```
2147                0                   0                            0
2148                1                   0                            0

      Forgetfulness  Diagnosis  DoctorInCharge
0                 0          0       XXXConfid
1                 1          0       XXXConfid
2                 0          0       XXXConfid
3                 0          0       XXXConfid
4                 0          0       XXXConfid
...             ...        ...              ...
2144              0          1       XXXConfid
2145              0          1       XXXConfid
2146              0          1       XXXConfid
2147              1          1       XXXConfid
2148              1          0       XXXConfid

[2149 rows x 35 columns]
```

```
# checking null values columns
df.isnull().sum()[df.isnull().sum() > 0]
```

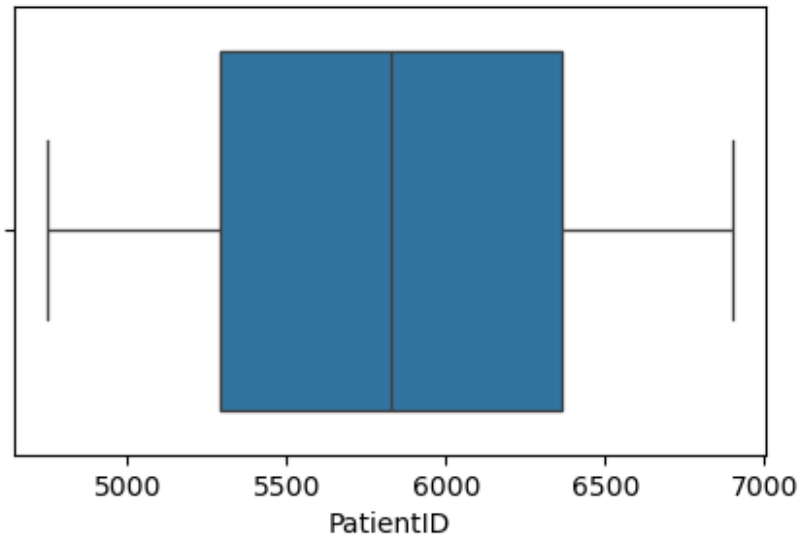
```
Series([], dtype: int64)
```

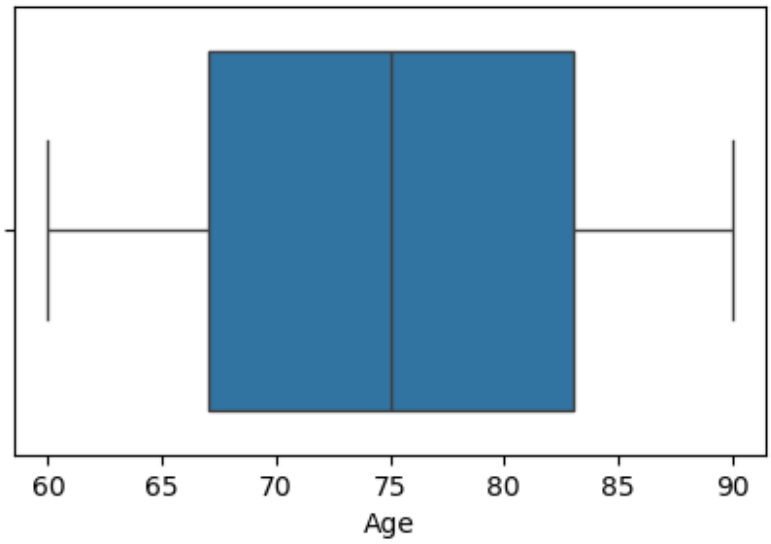
```
df.info()
```

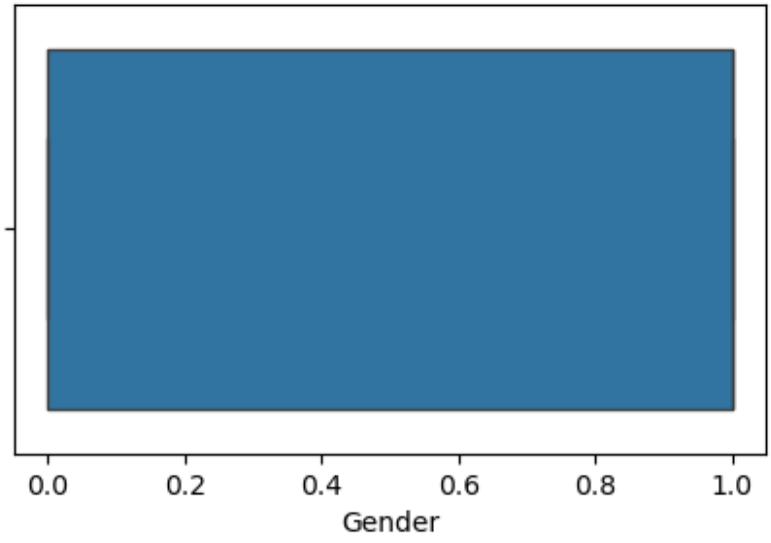
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2149 entries, 0 to 2148
Data columns (total 35 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   PatientID                  2149 non-null   int64
 1   Age                        2149 non-null   int64
 2   Gender                     2149 non-null   int64
 3   Ethnicity                  2149 non-null   int64
 4   EducationLevel             2149 non-null   int64
 5   BMI                        2149 non-null   float64
 6   Smoking                    2149 non-null   int64
 7   AlcoholConsumption         2149 non-null   float64
 8   PhysicalActivity           2149 non-null   float64
 9   DietQuality                2149 non-null   float64
 10  SleepQuality               2149 non-null   float64
 11  FamilyHistoryAlzheimers    2149 non-null   int64
 12  CardiovascularDisease      2149 non-null   int64
 13  Diabetes                   2149 non-null   int64
 14  Depression                 2149 non-null   int64
 15  HeadInjury                 2149 non-null   int64
 16  Hypertension               2149 non-null   int64
 17  SystolicBP                 2149 non-null   int64
 18  DiastolicBP                2149 non-null   int64
 19  CholesterolTotal           2149 non-null   float64
```

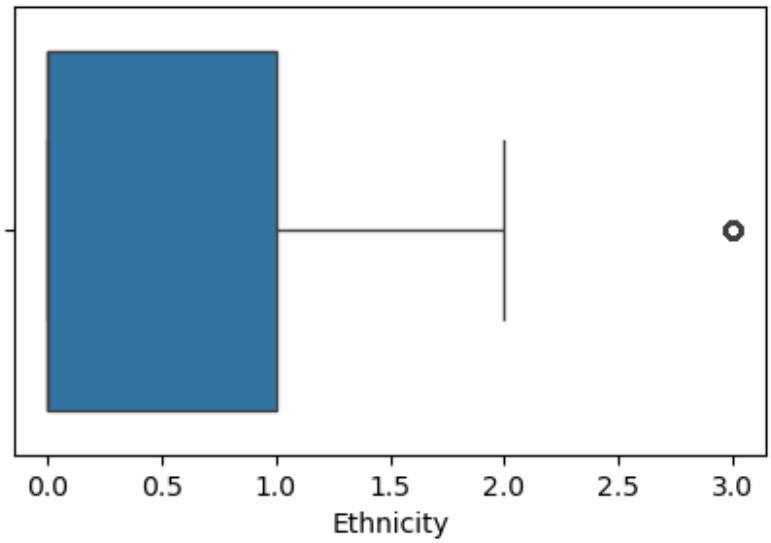
```
 20  CholesterolLDL              2149 non-null   float64
 21  CholesterolHDL              2149 non-null   float64
 22  CholesterolTriglycerides    2149 non-null   float64
 23  MMSE                        2149 non-null   float64
 24  FunctionalAssessment        2149 non-null   float64
 25  MemoryComplaints            2149 non-null   int64
 26  BehavioralProblems          2149 non-null   int64
 27  ADL                         2149 non-null   float64
 28  Confusion                   2149 non-null   int64
 29  Disorientation              2149 non-null   int64
 30  PersonalityChanges          2149 non-null   int64
 31  DifficultyCompletingTasks   2149 non-null   int64
 32  Forgetfulness               2149 non-null   int64
 33  Diagnosis                   2149 non-null   int64
 34  DoctorInCharge              2149 non-null   object
dtypes: float64(12), int64(22), object(1)
memory usage: 587.7+ KB


df['DoctorInCharge'].head(5)

0    XXXConfid
1    XXXConfid
2    XXXConfid
3    XXXConfid
4    XXXConfid
Name: DoctorInCharge, dtype: object

numerical = [i for i in df.columns if df[i].dtype in (['float64',
'int64'])]
numerical

['PatientID',
 'Age',
 'Gender',
 'Ethnicity',
 'EducationLevel',
 'BMI',
 'Smoking',
 'AlcoholConsumption',
 'PhysicalActivity',
 'DietQuality',
 'SleepQuality',
 'FamilyHistoryAlzheimers',
 'CardiovascularDisease',
 'Diabetes',
 'Depression',
 'HeadInjury',
 'Hypertension',
 'SystolicBP',
```

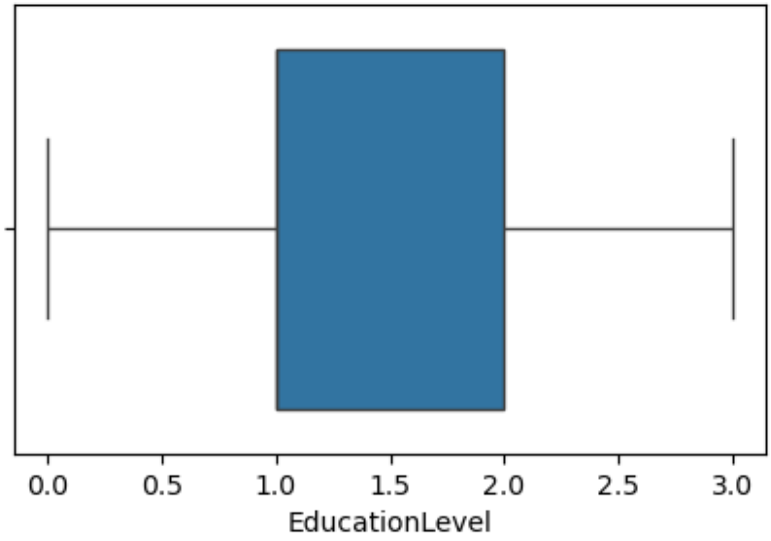
```
 'DiastolicBP',
 'CholesterolTotal',
 'CholesterolLDL',
 'CholesterolHDL',
 'CholesterolTriglycerides',
 'MMSE',
 'FunctionalAssessment',
 'MemoryComplaints',
 'BehavioralProblems',
 'ADL',
 'Confusion',
 'Disorientation',
 'PersonalityChanges',
 'DifficultyCompletingTasks',
 'Forgetfulness',
 'Diagnosis']

# detecting outliers
import matplotlib.pyplot as plt
import seaborn as sns

for i in numerical:
    plt.figure(figsize=(5,3))
    sns.boxplot(x=df[i])
    plt.show()
```

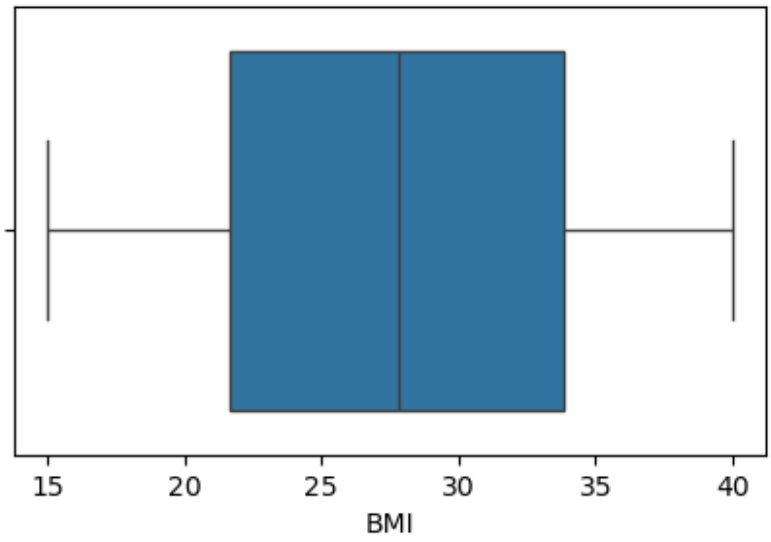


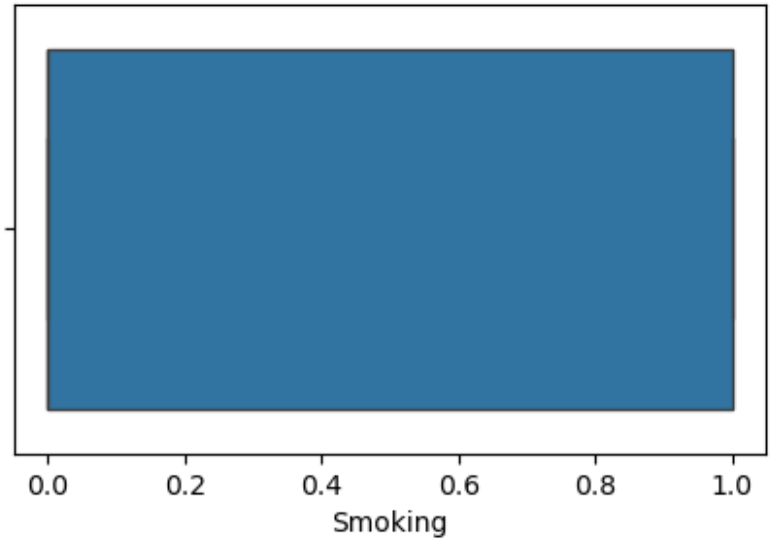

60
65
70
75
80
85
90
Age

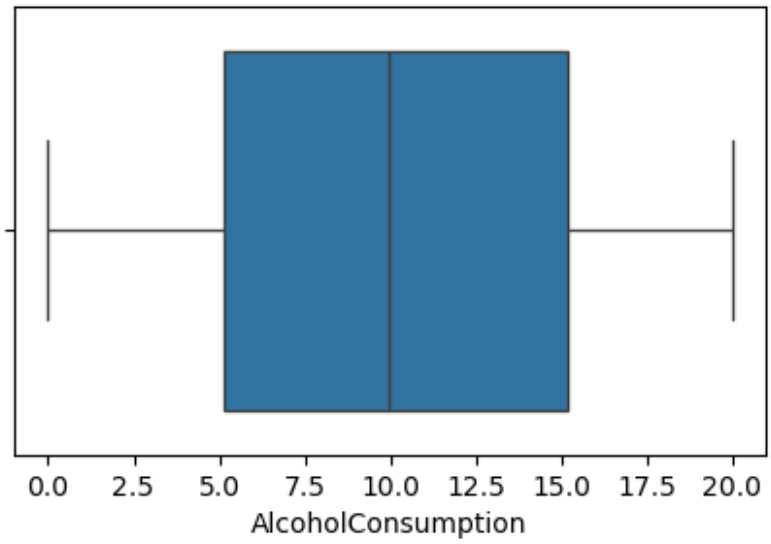

0.0
0.2
0.4
0.6
0.8
1.0
Gender

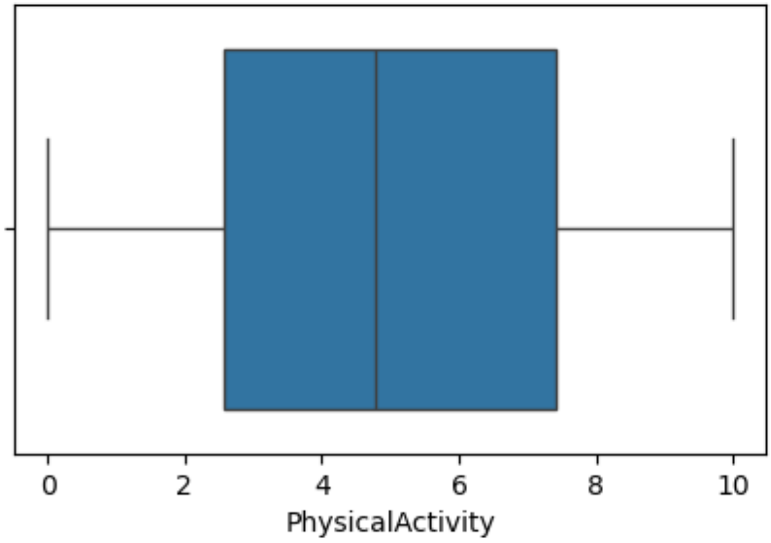

0.0
0.5
1.0
1.5
2.0
2.5
3.0
Ethnicity

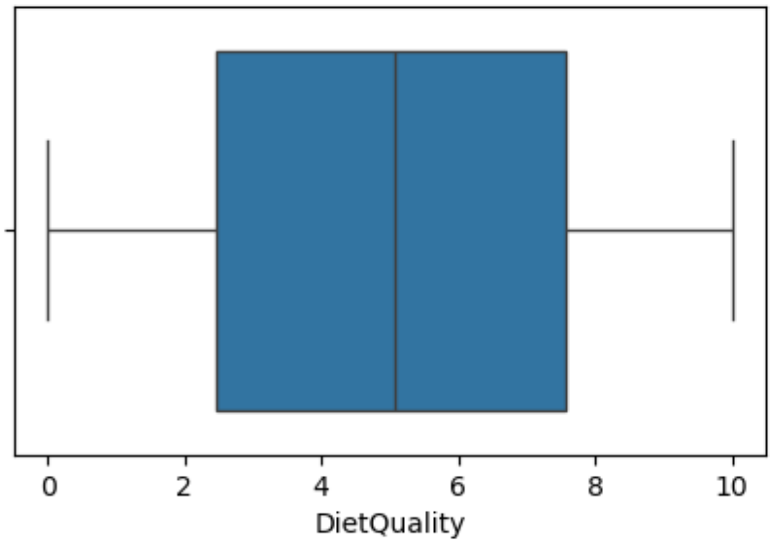

0.0
0.5
1.0
1.5
2.0
2.5
3.0
EducationLevel

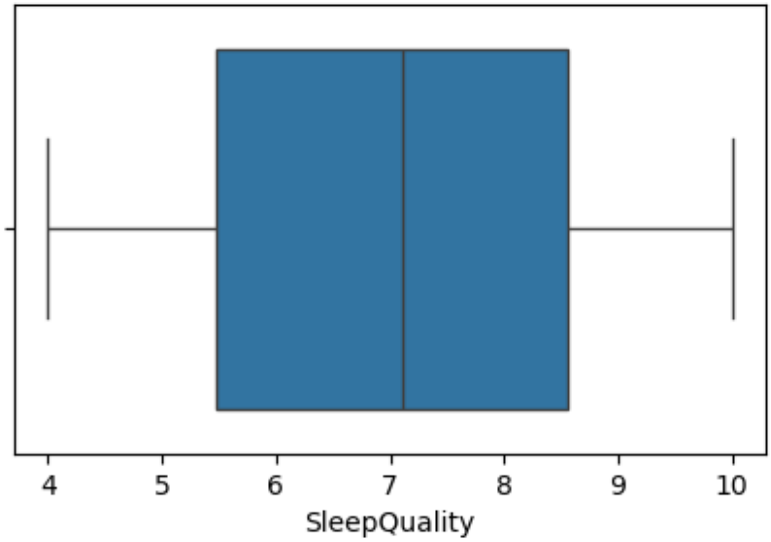

15
20
25
30
35
40
BMI

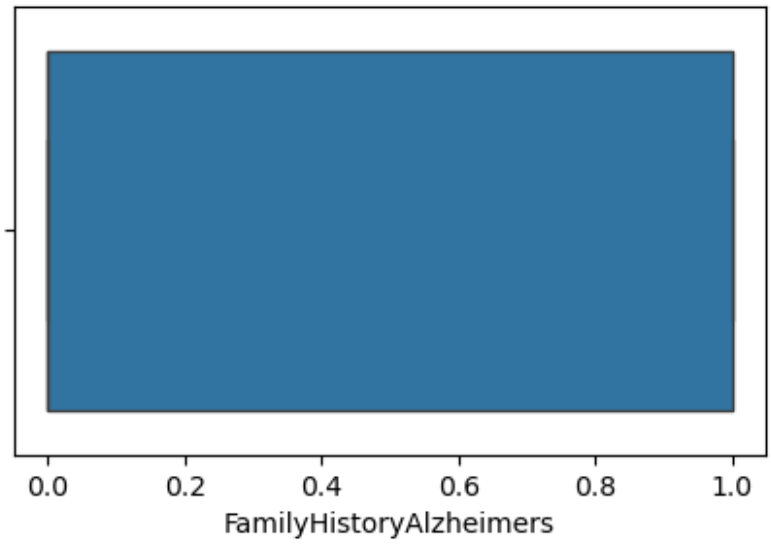

0.0
0.2
0.4
0.6
0.8
1.0
Smoking

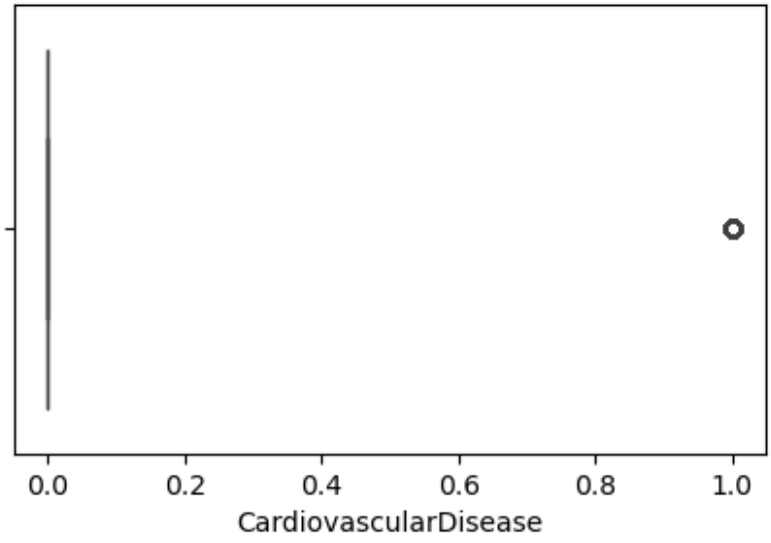

0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
AlcoholConsumption

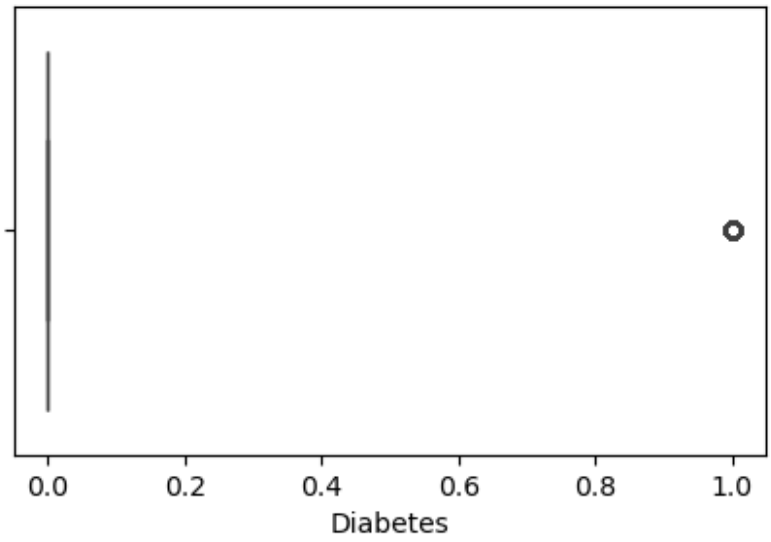

0
2
4
6
8
10
PhysicalActivity

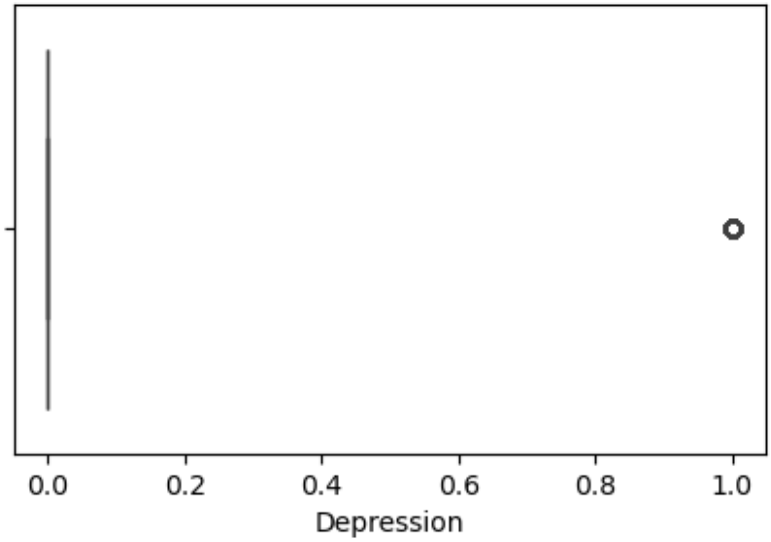

0
2
4
6
8
10
DietQuality

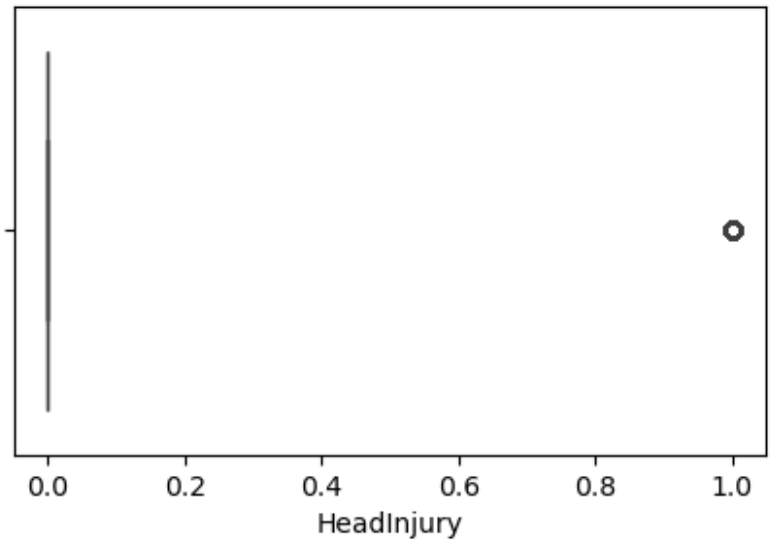

4
5
6
7
8
9
10
SleepQuality

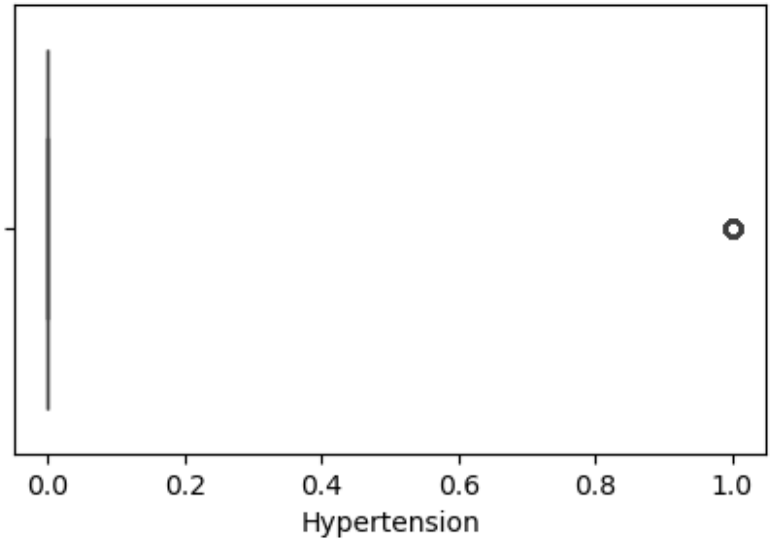

0.0
0.2
0.4
0.6
0.8
1.0
FamilyHistoryAlzheimers

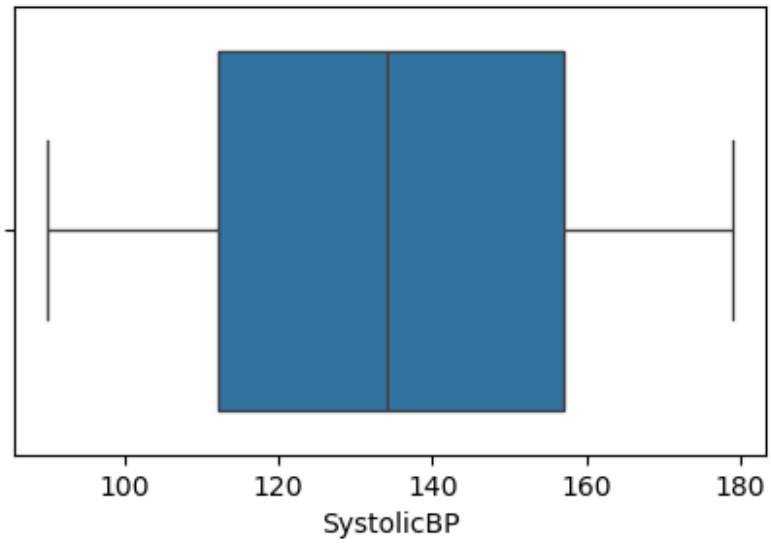

0.0
0.2
0.4
0.6
0.8
1.0
CardiovascularDisease

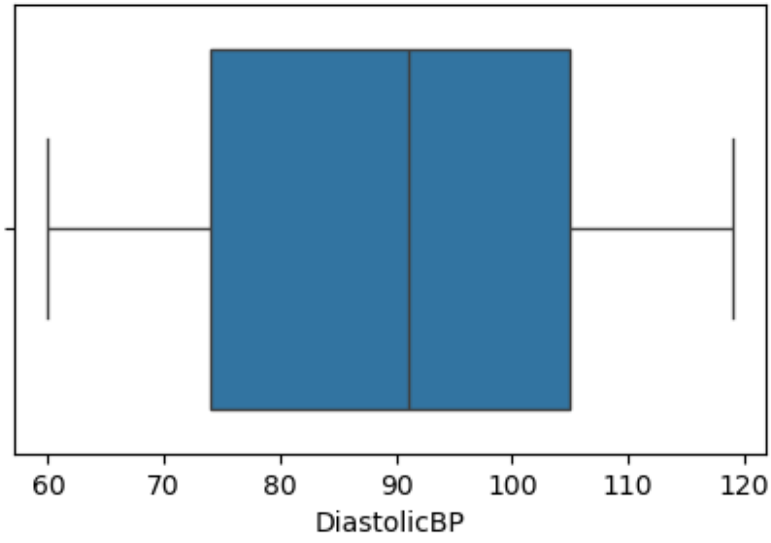

0.0
0.2
0.4
0.6
0.8
1.0
Diabetes

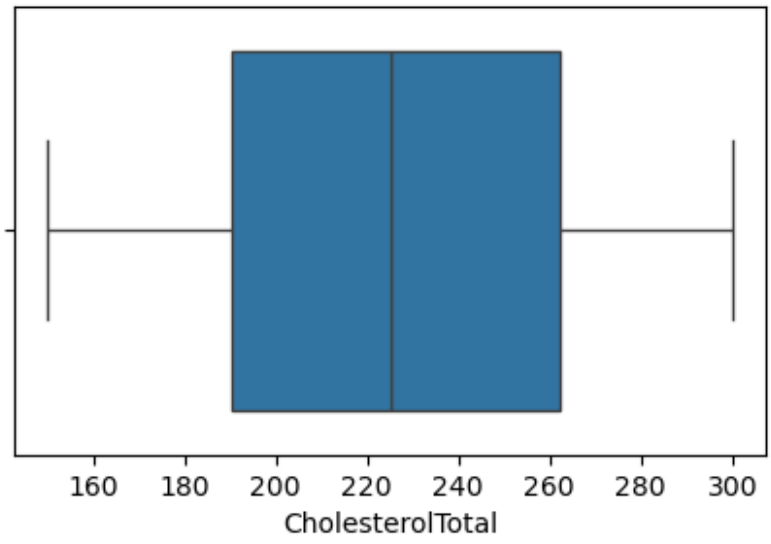

0.0
0.2
0.4
0.6
0.8
1.0
Depression

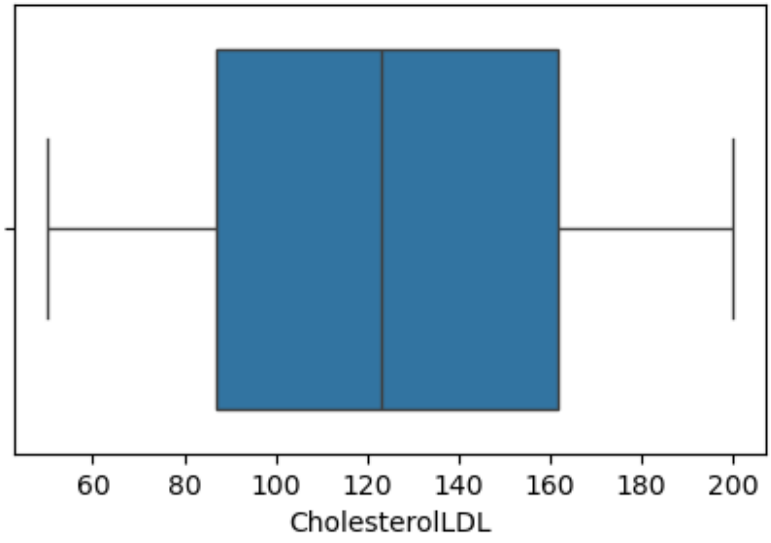

0.0
0.2
0.4
0.6
0.8
1.0
HeadInjury

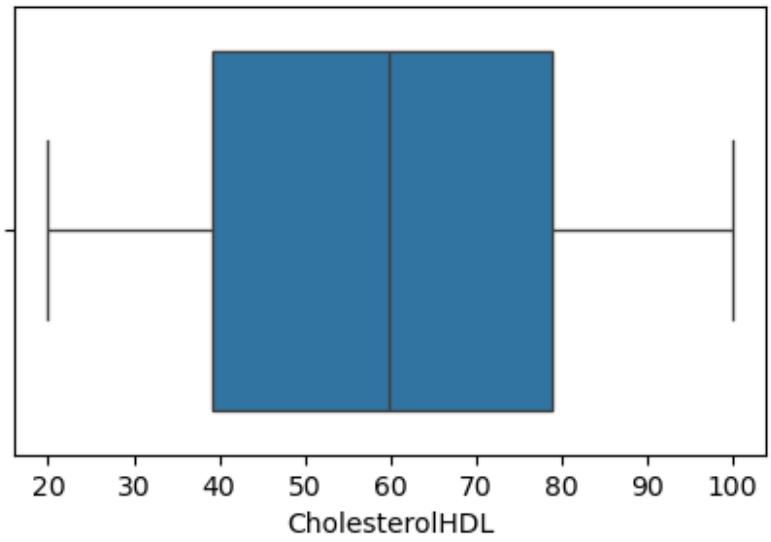

0.0
0.2
0.4
0.6
0.8
1.0
Hypertension

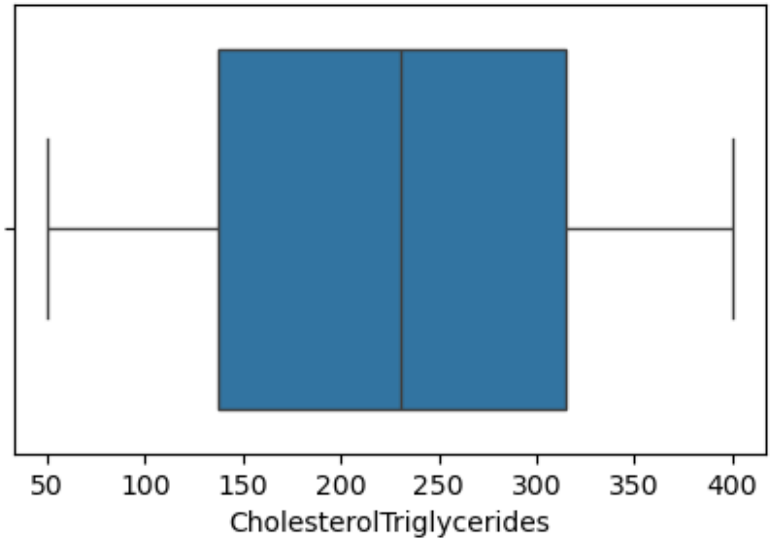

100
120
140
160
180
SystolicBP

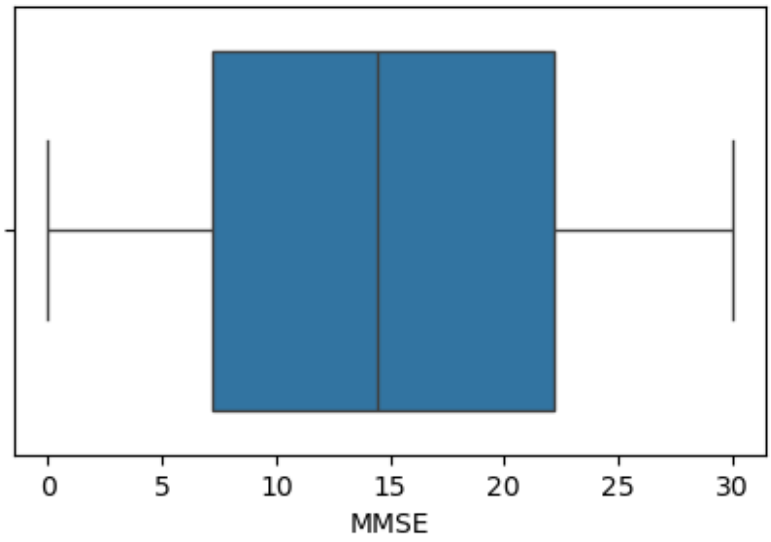

60
70
80
90
100
110
120
DiastolicBP

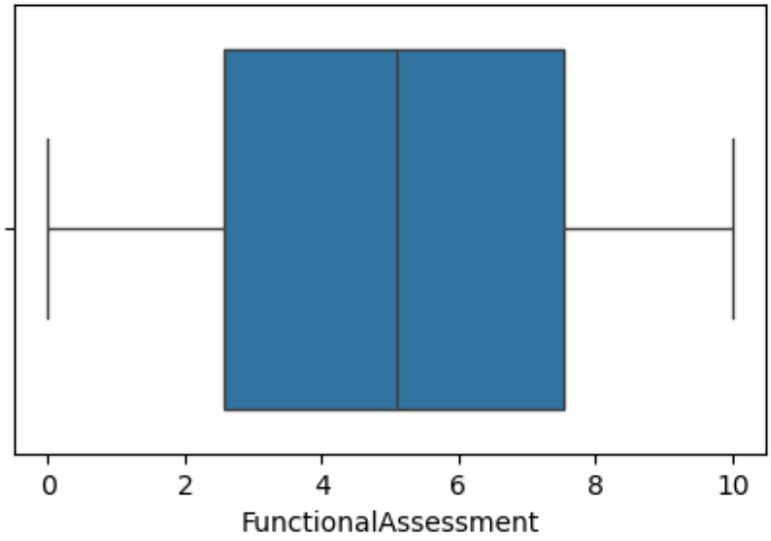

160
180
200
220
240
260
280
300
CholesterolTotal

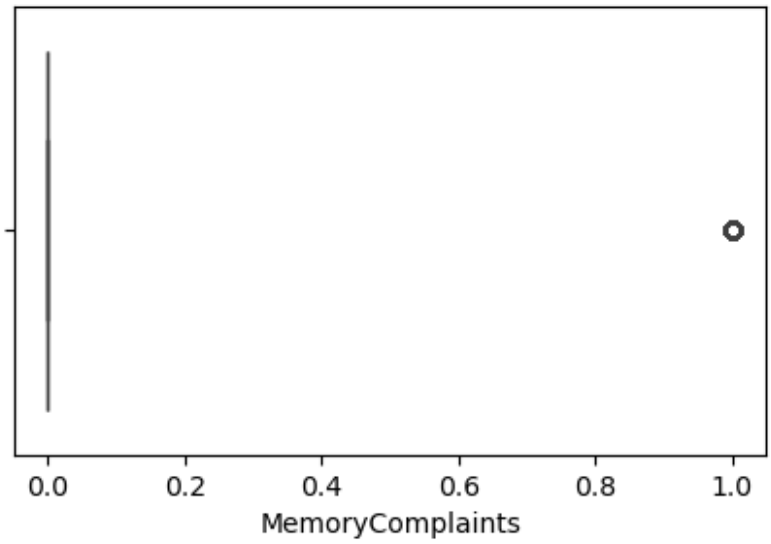

60
80
100
120
140
160
180
200
CholesterolLDL

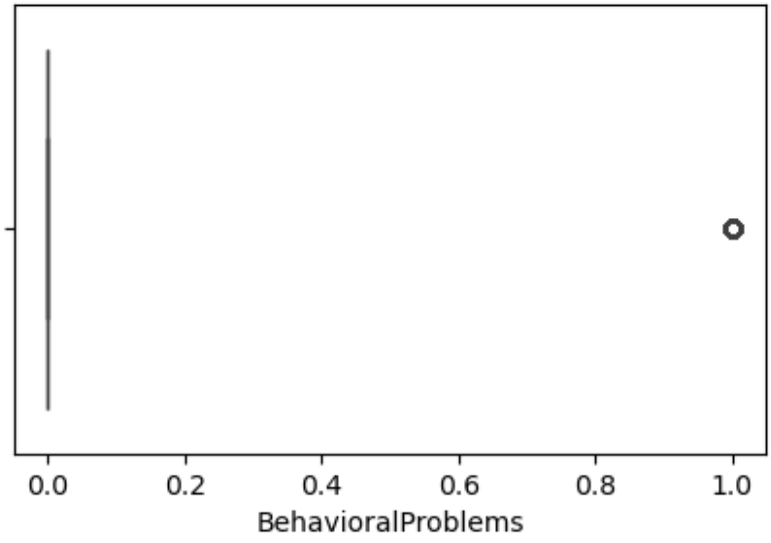

20
30
40
50
60
70
80
90
100
CholesterolHDL

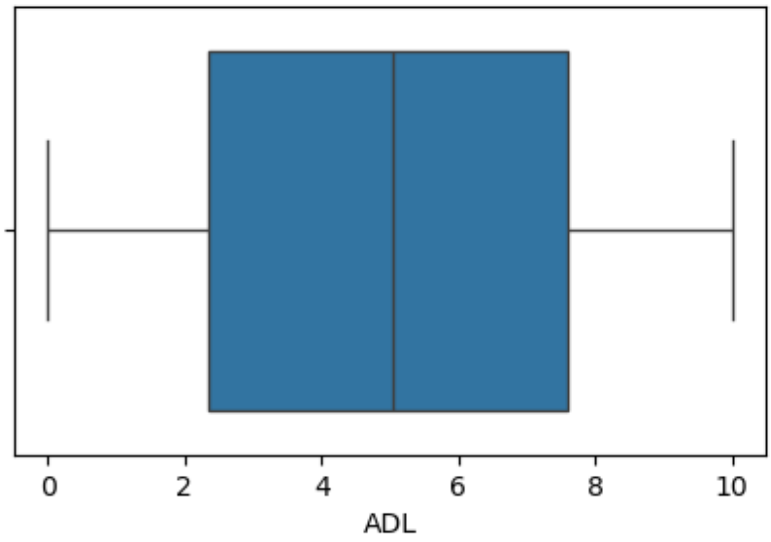

50
100
150
200
250
300
350
400
CholesterolTriglycerides

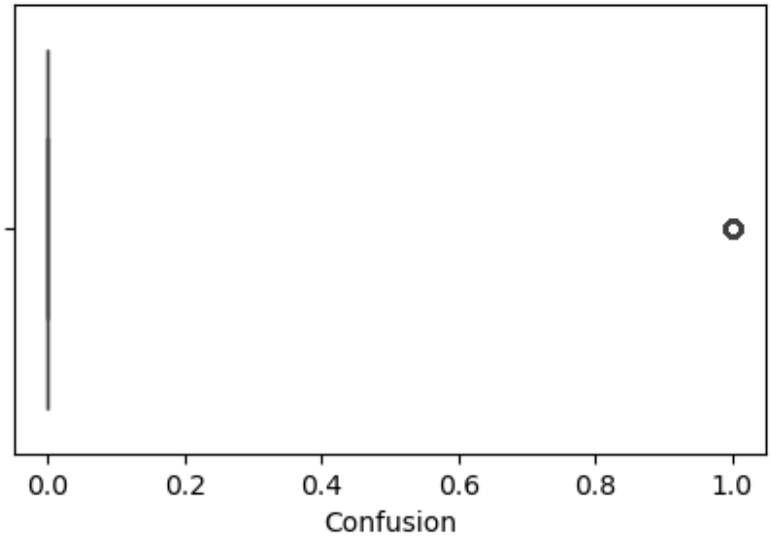

0
5
10
15
20
25
30
MMSE

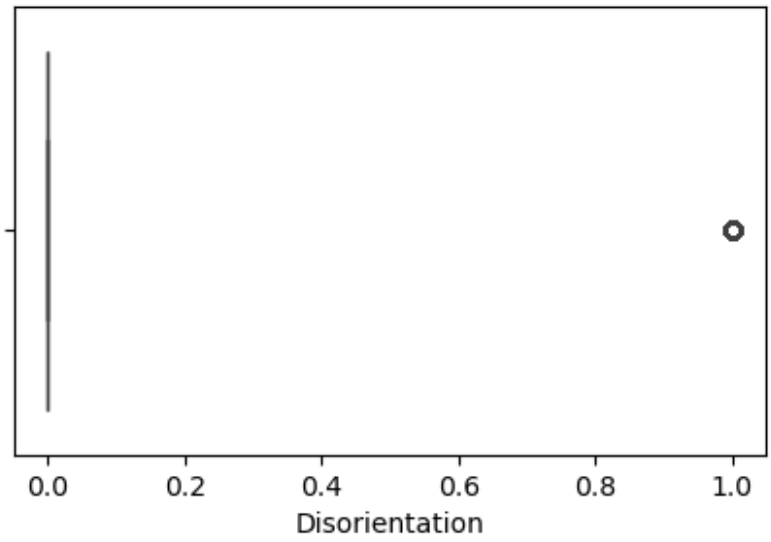

0
2
4
6
8
10
FunctionalAssessment

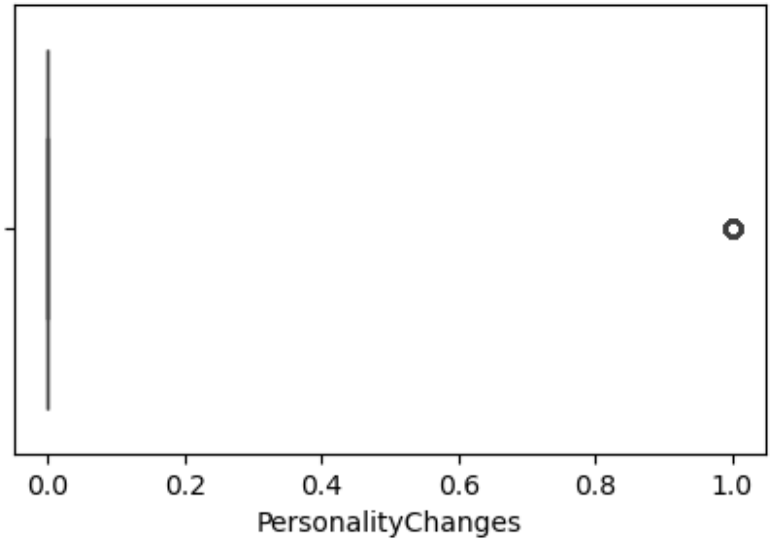

0.0
0.2
0.4
0.6
0.8
1.0
MemoryComplaints

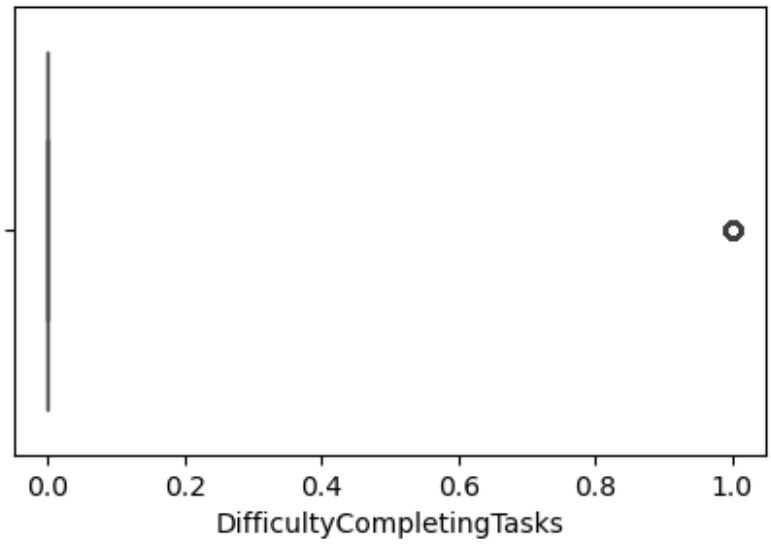

0.0
0.2
0.4
0.6
0.8
1.0
BehavioralProblems

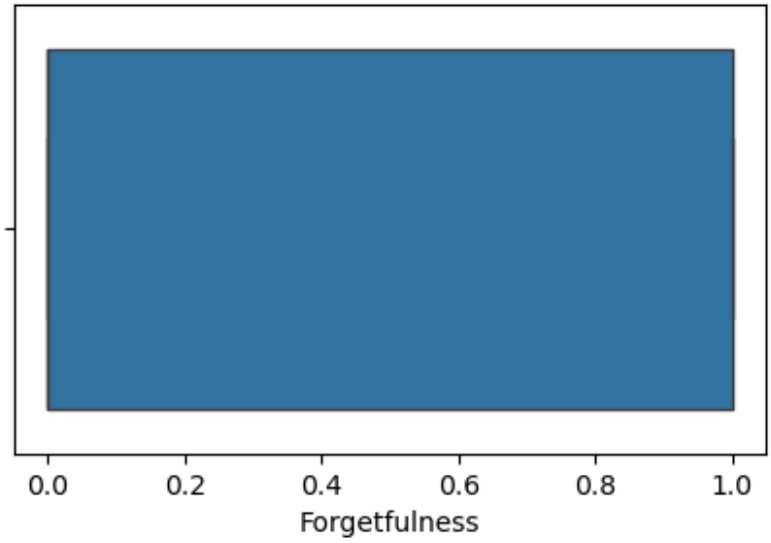

0
2
4
6
8
10
ADL

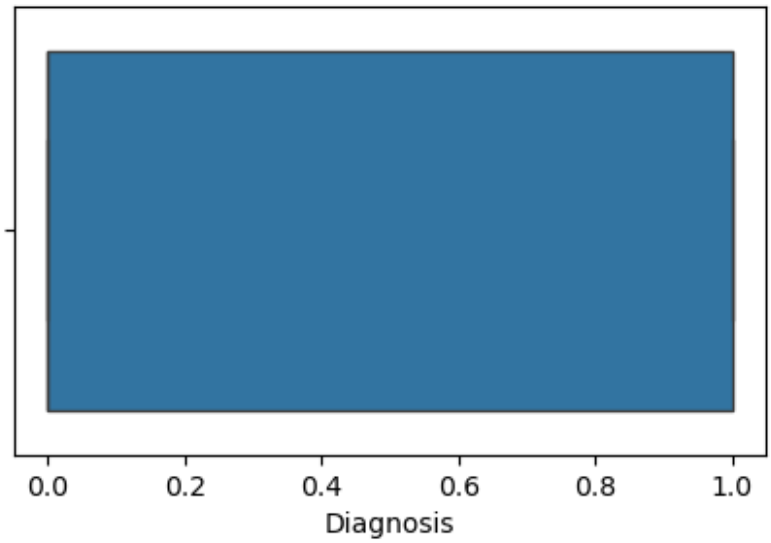

0.0
0.2
0.4
0.6
0.8
1.0
Confusion


0.0
0.2
0.4
0.6
0.8
1.0
Disorientation


0.0
0.2
0.4
0.6
0.8
1.0
PersonalityChanges


0.0
0.2
0.4
0.6
0.8
1.0
DifficultyCompletingTasks


0.0
0.2
0.4
0.6
0.8
1.0
Forgetfulness



Checked numerical columns using boxplots. No significant outliers found, so no treatment needed. Data is clean and ready for model training.

```
df[numerical].info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2149 entries, 0 to 2148
Data columns (total 34 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   PatientID                  2149 non-null   int64  
 1   Age                        2149 non-null   int64  
 2   Gender                     2149 non-null   int64  
 3   Ethnicity                  2149 non-null   int64  
 4   EducationLevel             2149 non-null   int64  
 5   BMI                        2149 non-null   float64
 6   Smoking                    2149 non-null   int64  
 7   AlcoholConsumption         2149 non-null   float64
 8   PhysicalActivity           2149 non-null   float64
 9   DietQuality                2149 non-null   float64
 10  SleepQuality               2149 non-null   float64
 11  FamilyHistoryAlzheimers    2149 non-null   int64  
 12  CardiovascularDisease      2149 non-null   int64  
 13  Diabetes                   2149 non-null   int64  
 14  Depression                 2149 non-null   int64  
 15  HeadInjury                 2149 non-null   int64  
 16  Hypertension               2149 non-null   int64  
 17  SystolicBP                 2149 non-null   int64  
 18  DiastolicBP                2149 non-null   int64  
 19  CholesterolTotal           2149 non-null   float64
 20  CholesterolLDL             2149 non-null   float64
 21  CholesterolHDL             2149 non-null   float64
```

```
 22  CholesterolTriglycerides   2149 non-null   float64
 23  MMSE                       2149 non-null   float64
 24  FunctionalAssessment       2149 non-null   float64
 25  MemoryComplaints           2149 non-null   int64
 26  BehavioralProblems         2149 non-null   int64
 27  ADL                        2149 non-null   float64
 28  Confusion                  2149 non-null   int64
 29  Disorientation             2149 non-null   int64
 30  PersonalityChanges         2149 non-null   int64
 31  DifficultyCompletingTasks  2149 non-null   int64
 32  Forgetfulness              2149 non-null   int64
 33  Diagnosis                  2149 non-null   int64
dtypes: float64(12), int64(22)
memory usage: 571.0 KB
```

```
# detecting muticolinearity

from statsmodels.stats.outliers_influence import
variance_inflation_factor
import statsmodels.api as sm
import pandas as pd

x_constant = sm.add_constant(df[numerical]) # this will add constant
column

# priting vif table to check multicollinearity

vif_data = pd.DataFrame()
vif_data['Features'] = x_constant.columns
vif_data['data'] = [variance_inflation_factor(x_constant.values, i)
for i in range(x_constant.shape[1])]
print(vif_data)
```

```
                     Features        data
0                       const  335.143471
1                   PatientID    1.015420
2                         Age    1.018495
3                      Gender    1.012813
4                   Ethnicity    1.011026
5              EducationLevel    1.018564
6                         BMI    1.017413
7                     Smoking    1.015848
8          AlcoholConsumption    1.009480
9            PhysicalActivity    1.009126
10                DietQuality    1.015186
11               SleepQuality    1.016777
12    FamilyHistoryAlzheimers    1.011505
13      CardiovascularDisease    1.014498
14                   Diabetes    1.014428
15                 Depression    1.010652
```

```
16                  HeadInjury    1.015184
17               Hypertension    1.016088
18                 SystolicBP    1.009065
19                DiastolicBP    1.009712
20           CholesterolTotal    1.011762
21             CholesterolLDL    1.014931
22             CholesterolHDL    1.014014
23   CholesterolTriglycerides    1.015984
24                       MMSE    1.106521
25       FunctionalAssessment    1.222205
26           MemoryComplaints    1.171280
27         BehavioralProblems    1.119832
28                        ADL    1.189009
29                  Confusion    1.009375
30             Disorientation    1.017712
31         PersonalityChanges    1.013733
32  DifficultyCompletingTasks    1.019379
33              Forgetfulness    1.014404
34                  Diagnosis    1.781116
```

```
# filtering the vif over 5
vif_data[(vif_data['data'] > 5) &
(~vif_data['Features'].isin(['const', 'Diagnosis']))]
```

```
Empty DataFrame
Columns: [Features, data]
Index: []
```

```
# checking correlation with Diagnosis column

df[numerical].corr()['Diagnosis'].sort_values(ascending=False)
```

```
Diagnosis                   1.000000
MemoryComplaints            0.306742
BehavioralProblems          0.224350
CholesterolHDL              0.042584
PatientID                   0.041019
Hypertension                0.035080
CardiovascularDisease       0.031490
BMI                         0.026343
CholesterolTriglycerides    0.022672
DifficultyCompletingTasks   0.009069
DietQuality                 0.008506
CholesterolTotal            0.006394
PhysicalActivity            0.005945
DiastolicBP                 0.005293
Forgetfulness              -0.000354
Smoking                    -0.004865
Age                        -0.005488
Depression                 -0.005893
```

```
AlcoholConsumption           -0.007618
Ethnicity                    -0.014782
SystolicBP                   -0.015615
Confusion                    -0.019186
PersonalityChanges           -0.020627
Gender                       -0.020975
HeadInjury                   -0.021411
Disorientation               -0.024648
Diabetes                     -0.031508
CholesterolLDL               -0.031976
FamilyHistoryAlzheimers      -0.032900
EducationLevel               -0.043966
SleepQuality                 -0.056548
MMSE                         -0.237126
ADL                          -0.332346
FunctionalAssessment         -0.364898
Name: Diagnosis, dtype: float64
```

Checked multicolinearity and correlation, so multicolinearity and no strong correlation found. Data is ready for logistic regression

```
# sepearting independant and dependant variables
x = df[numerical].drop('Diagnosis',axis=1)
y = df['Diagnosis']

# I will split the data into 70% training 30 % testing

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.3, random_state=42)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1504, 33)
(645, 33)
(1504,)
(645,)
```

```
## training the logistic model

from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

```
# StandardScaler makes all values in columns similar in size so that one big column does not dominate the smaller columns. Everthing will be fair
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

model = LogisticRegression(class_weight='balanced', max_iter=2000) # class_weight='balanced' make balance between the categories.
'''Like if one category has 500 data and the other category has 100 data, without class weight balance, model will learn more from the category which has 500 rows as it has more data.. So after using balance, more weightage will be given to smaller class '''

model.fit(x_train,y_train)

LogisticRegression(class_weight='balanced', max_iter=2000)

## How scaler works
Before using scaler:

| A   | B  |
| --- | -- |
| 500 | 10 |
| 600 | 20 |
| 700 | 30 |

after using scaler:

| A (scaled) | B (scaled) |
| ---------- | ---------- |
| -1.0       | -1.0       |
| 0.0        | 0.0        |
| +1.0       | +1.0       |

Now both columns are in a similar range (-1 to +1).

## scaler(z) = x - mean / std ##
```

Note: What class_weight='balanced' Does

It automatically gives more importance to the smaller class and less to the larger class.

This helps the model not ignore the minority class (like Alzheimer's patients).

It balances the learning so both classes are treated fairly.

Result → improves recall for the smaller class.

```
print(model.score(x_train,y_train))
print(model.score(x_test,y_test))
```

```
0.8430851063829787
0.8031007751937984
```

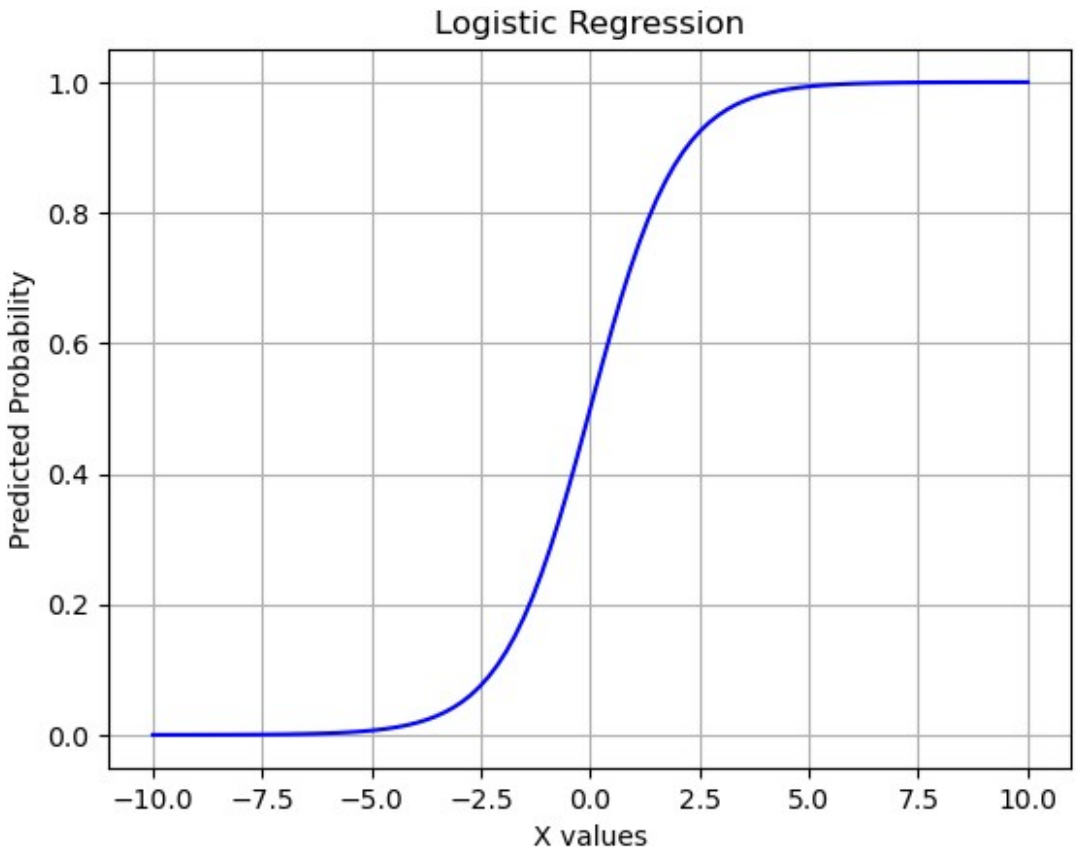
```
# lets check can it predit well or not

y_predict = model.predict(x_test)

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

accuracy_score = accuracy_score(y_test, y_predict)
confusion_matrix = confusion_matrix(y_test, y_predict)
classification_report = classification_report(y_test, y_predict)

print(f'accuracy_score is {accuracy_score}')
print(f'confusion_matrix is {confusion_matrix}')
print(f'classification_report is {classification_report}')
```

```
accuracy_score is 0.8031007751937984
confusion_matrix is [[327  74]
 [ 53 191]]
classification_report is               precision    recall  f1-score
support

           0       0.86      0.82      0.84       401
           1       0.72      0.78      0.75       244

    accuracy                           0.80       645
   macro avg       0.79      0.80      0.79       645
weighted avg       0.81      0.80      0.80       645
```

```
# My model predicted80% accuracy after using StandardScaler and
class_weight= 'balanced'
```

```
|                | **Predicted = 0**         | **Predicted = 1**
|
| -------------- | ------------------------- |
------------------------- |
| **Actual = 0** | □ **True Negative (TN)**  | □ **False Positive
(FP)** |
| **Actual = 1** | □ **False Negative (FN)** | □ **True Positive
(TP)**  |

□ Rows = Actual values
□ Columns = Predicted values
```

```
My confusion matrix:

[[327  74]
 [ 53 191]]
|                 | **Predicted = 0** | **Predicted = 1** |
| --------------- | ----------------- | ----------------- |
| **Actual = 0** | **327 → TN**       | **74 → FP**       |
| **Actual = 1** | **53 → FN**       | **191 → TP**      |

| Term                     | What it means
| Example                                        |
| ------------------------ |
------------------------------------------------------------------- |
---------------------------------------------- |
| **TN (True Negative)**  | Model correctly said “No Alzheimer’s”
| 327 people truly don’t have it, model said no |
| **FP (False Positive)** | Model was wrong — said “Has Alzheimer’s”
but person is healthy     | 74 people
|
| **FN (False Negative)** | Model was wrong — said “No Alzheimer’s”
but person actually has it | 53 people
|
| **TP (True Positive)**  | Model correctly said “Has Alzheimer’s”
| 191 people                                      |


# My F1 score show an average of 80% pristive between patient wha has
Alzheimer's or who has not.

## plotting my model

import numpy as np
import matplotlib.pyplot as plt

# x-axis values
x = np.linspace(-10, 10, 100) # it will give me 100 equal values
between -10 to +10

# sigmoid formula
y = 1 / (1 + np.exp(-x)) # no.exp is the numpy exponention function

# plot the curve
plt.plot(x, y, color='blue')
plt.title('Logistic Regression')
plt.xlabel('X values')
plt.ylabel('Predicted Probability')
plt.grid(True)
plt.show()
```



In np.linspace(-10, 10, 100), I took −10 to +10 so the sigmoid curve fully covers the range where probabilities go from near 0 to near 1.