MAKERERE UNIVERSITY

SEMESTER ONE 2024/2025 ACADEMIC YEAR

SCHOOL OF COMPUTING AND INFORMATICS TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

MASTER OF SCIENCE IN COMPUTER SCIENCE

MCS 7103

MACHINE LEARNING

ASSIGNMENT ONE

AGABA LUCKY

2024/HD05/21913U

2400721913

This dataset is from the National Poll on Healthy Aging (NPHA) filtered down to develop and validate machine learning algorithms for predicting the number of doctors a survey respondent sees in a year. This dataset's records represent seniors who responded to the NPHA survey.

The dataset is a tabular dataset with the subject area of Health and Medicine associated with classification. It is a definite feature type with 714 instances and 14 features.

My Question before;

How can I explore and understand the National Poll on Healthy Aging data to effectively prepare it for machine learning analysis and which problem am I handling in particular?

Answer; I can achieve this by performing Exploratory Data Analysis (EDA) on this dataset and the problem being handled is predicting the number of doctors a survey respondent sees in a year to age or grow old when he or she is healthy.

My Questions after;

1. *What are the key features of my dataset, and what types of data are included?*

Answer; The dataset includes features such as age, gender, health conditions, and lifestyle factors.

2. *Are there missing values or data quality issues that need to be addressed?*

Answer; There are no missing values but some issues need to be addressed.

3. *What is my target Variable?*

Answer; The number of doctors visited is my target variable.

4. *How many instances and features do I have and are they really good enough for my dataset?*

Answer; I have 714 instances and 14 features and they are good enough to have the prediction achieved.

Below is a breakdown of the variables that were used;

Variable Name	**Role**	**Type**	**Description**
Number_of_Doctors_Visited	Target	Categorical	The total count of different doctors the patient has seen = { 1: 0-1 doctors 2: 2-3 doctors 3: 4 or more doctors }
Age	Feature	Categorical	The patient's age group = { 1: 50-64 2: 65-80 }

Physical_Health	Feature	Categorical	A self-assessment of the patient's physical well-being = { -1: Refused 1: Excellent 2: Very Good 3: Good 4: Fair 5: Poor }
Mental_Health	Feature	Categorical	A self-evaluation of the patient's mental or psychological health = { -1: Refused 1: Excellent 2: Very Good 3: Good 4: Fair 5: Poor }
Dental_Health	Feature	Categorical	A self-assessment of the patient's oral or dental health= { -1: Refused 1: Excellent 2: Very Good 3: Good 4: Fair 5: Poor }
Employment	Feature	Categorical	The patient's employment status or work-related information = { -1: Refused 1: Working full-time 2: Working part-time 3: Retired 4: Not working at this time }
Stress_Keeps_Patient_from_Sleeping	Feature	Categorical	Whether stress affects the patient's ability to sleep = { 0: No 1: Yes }
Medication_Keeps_Patient_from_Sleeping	Feature	Categorical	Whether medication impacts the patient's sleep = { 0: No 1: Yes }
Pain_Keeps_Patient_from_Sleeping	Feature	Categorical	Whether physical pain disturbs the patient's sleep = { 0: No 1: Yes }
Bathroom_Needs_Keeps_Patient_from_Sleeping	Feature	Categorical	Whether the need to use the bathroom affects the patient's sleep = { 0: No 1: Yes }
Uknown_Keeps_Patient_from_Sleeping	Feature	Categorical	Unidentified factors affecting the patient's sleep = { 0: No 1: Yes }
Trouble_Sleeping	Feature	Categorical	General issues or difficulties the patient

			faces with sleeping = { 0: No 1: Yes }
Prescription_Sleep_Medication	Feature	Categorical	Information about any sleep medication prescribed to the patient = { -1: Refused 1: Use regularly 2: Use occasionally 3: Do not use }
Race	Feature	Categorical	The patient's racial or ethnic background = { -2: Not asked -1: REFUSED 1: White, Non-Hispanic 2: Black, Non-Hispanic 3: Other, Non-Hispanic 4: Hispanic 5: 2+ Races, Non-Hispanic }
Gender	Feature	Categorical	The gender identity of the patient = { -2: Not asked -1: REFUSED 1: Male 2: Female }

Importing the necessary libraries;

I imported all the necessary libraries that I needed to use while working with my dataset as follows.

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

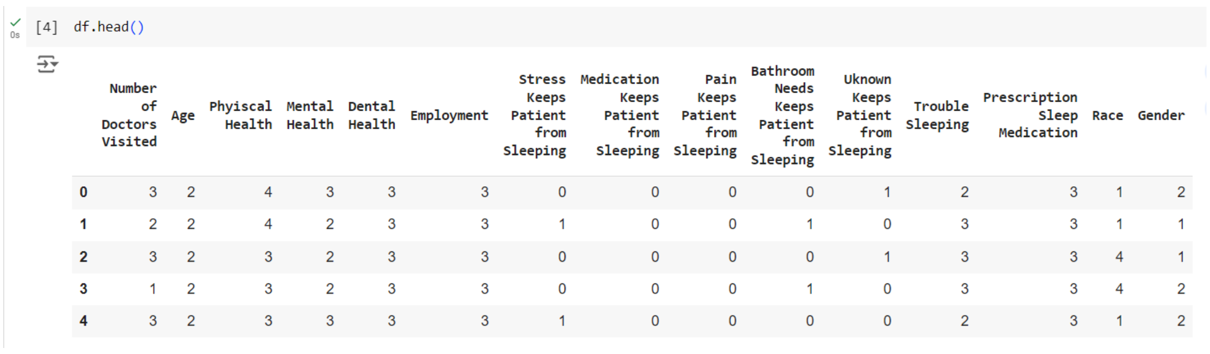
Reading the dataset;

I loaded my dataset into Google Collab and this was achieved successfully.

```
df = pd.read_csv('/content/NPHA-doctor-visits.csv')
```

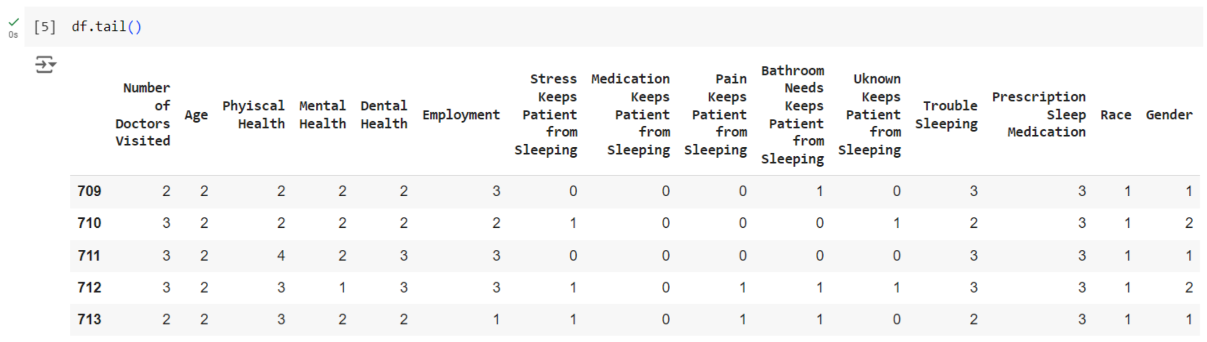
Performing a sanity check on the data;

I performed a sanity check on my dataset by finding out the first five and last five rows of my dataset.



```
[4] df.head()
```

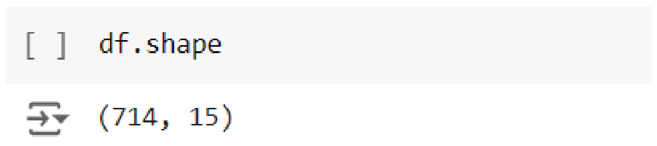
	Number of Doctors Visited	Age	Phyiscal Health	Mental Health	Dental Health	Employment	Stress Keeps Patient from Sleeping	Medication Keeps Patient from Sleeping	Pain Keeps Patient from Sleeping	Bathroom Needs Keeps Patient from Sleeping	Uknown Keeps Patient from Sleeping	Trouble Sleeping	Prescription Sleep Medication	Race	Gender
0	3	2	4	3	3	3	0	0	0	0	1	2	3	1	2
1	2	2	4	2	3	3	1	0	0	1	0	3	3	1	1
2	3	2	3	2	3	3	0	0	0	0	1	3	3	4	1
3	1	2	3	2	3	3	0	0	0	1	0	3	3	4	2
4	3	2	3	3	3	3	1	0	0	0	0	2	3	1	2



```
[5] df.tail()
```

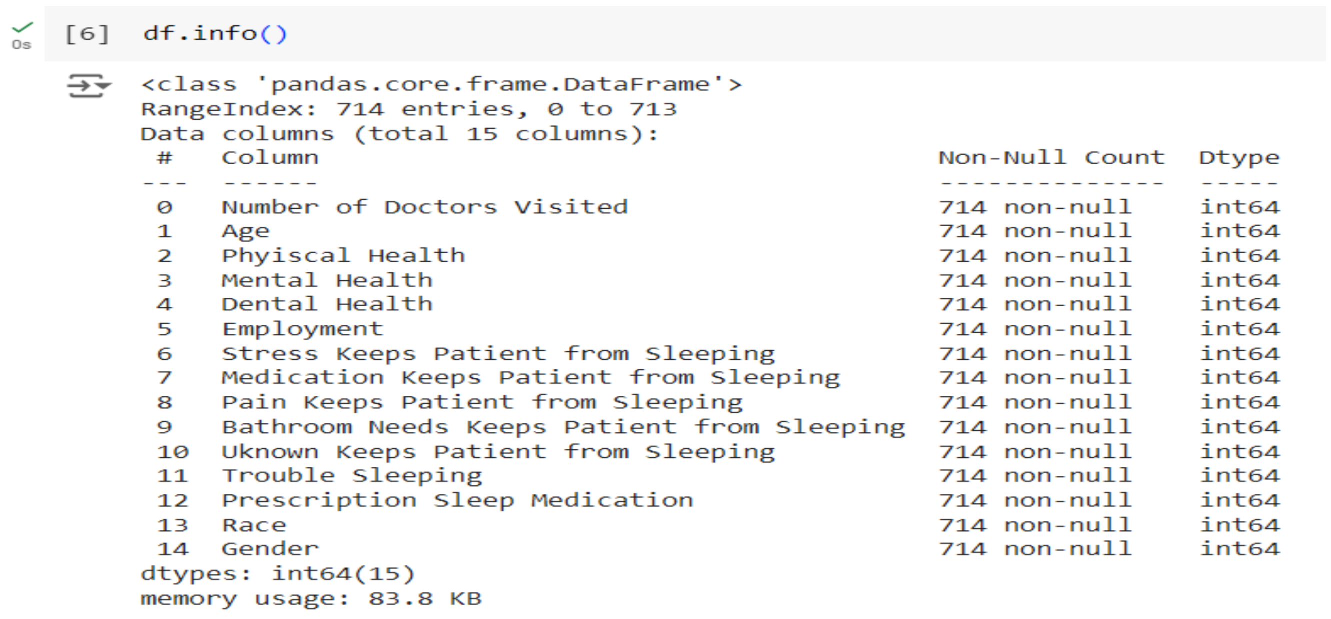
	Number of Doctors Visited	Age	Phyiscal Health	Mental Health	Dental Health	Employment	Stress Keeps Patient from Sleeping	Medication Keeps Patient from Sleeping	Pain Keeps Patient from Sleeping	Bathroom Needs Keeps Patient from Sleeping	Uknown Keeps Patient from Sleeping	Trouble Sleeping	Prescription Sleep Medication	Race	Gender
709	2	2	2	2	2	3	0	0	0	1	0	3	3	1	1
710	3	2	2	2	2	2	1	0	0	0	1	2	3	1	2
711	3	2	4	2	3	3	0	0	0	0	0	3	3	1	1
712	3	2	3	1	3	3	1	0	1	1	1	3	3	1	2
713	2	2	3	2	2	1	1	0	1	1	0	2	3	1	1

I went ahead to find out the number of rows and columns that I had in my dataset and I discovered that I had 714 rows and 15 columns.



```
[ ] df.shape

(714, 15)
```

The next was to see the columns and their data types, for this, I used the *info()* method.

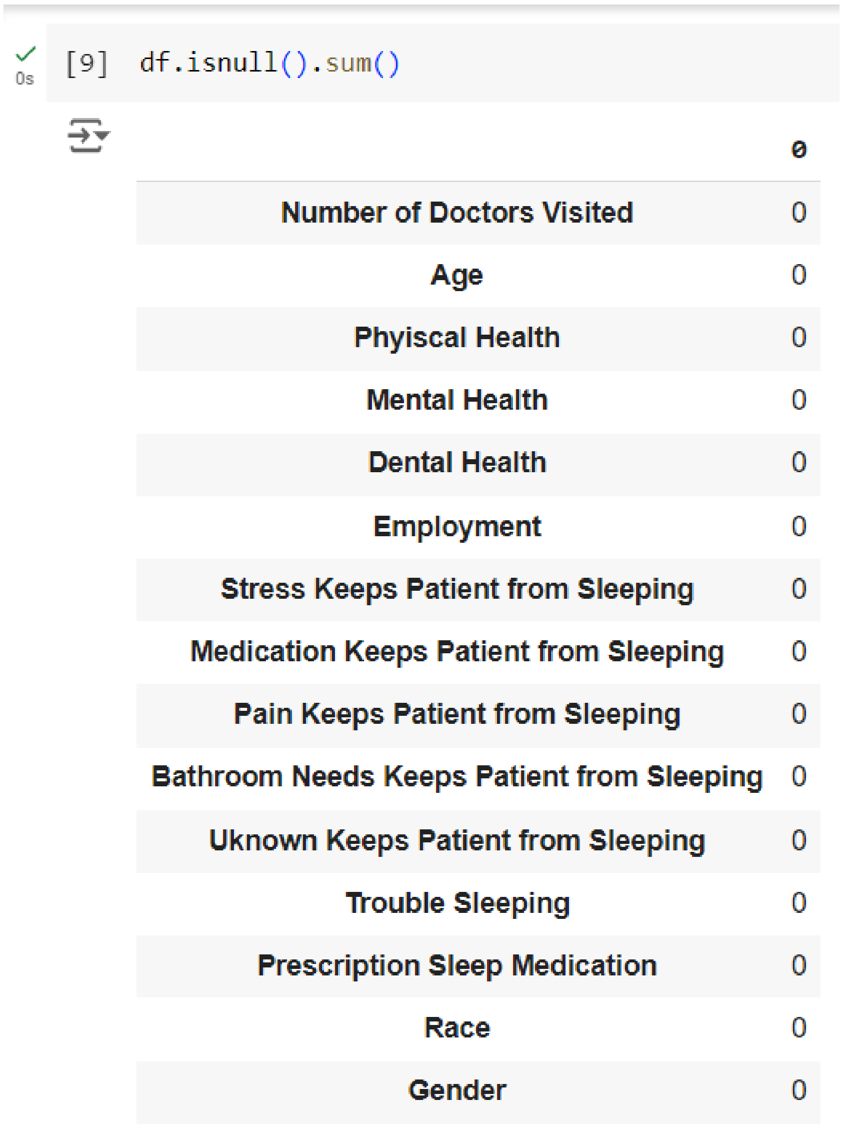


```
[6] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 714 entries, 0 to 713
Data columns (total 15 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   Number of Doctors Visited                   714 non-null    int64
 1   Age                                         714 non-null    int64
 2   Phyiscal Health                             714 non-null    int64
 3   Mental Health                               714 non-null    int64
 4   Dental Health                               714 non-null    int64
 5   Employment                                  714 non-null    int64
 6   Stress Keeps Patient from Sleeping          714 non-null    int64
 7   Medication Keeps Patient from Sleeping      714 non-null    int64
 8   Pain Keeps Patient from Sleeping            714 non-null    int64
 9   Bathroom Needs Keeps Patient from Sleeping  714 non-null    int64
 10  Uknown Keeps Patient from Sleeping          714 non-null    int64
 11  Trouble Sleeping                            714 non-null    int64
 12  Prescription Sleep Medication               714 non-null    int64
 13  Race                                        714 non-null    int64
 14  Gender                                      714 non-null    int64
dtypes: int64(15)
memory usage: 83.8 KB
```

I also found out the number of unique elements in my dataset using the *nunique()* method (*df.nunique()*). This helped me to decide which type of encoding to choose for converting categorical columns into numerical columns in case of any.

Handling missing values;

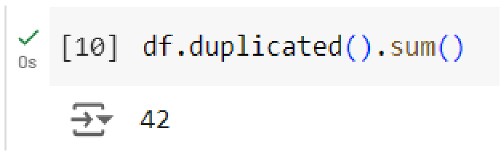
In checking for missing values, the *df.isnull().sum*() method was used, and it was realized that there were no missing values in my dataset.



```
[9] df.isnull().sum()
```

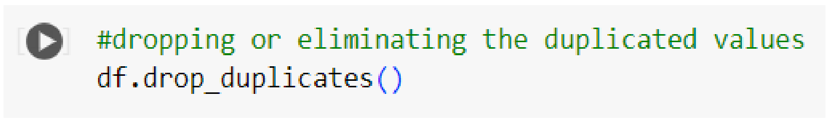
	0
Number of Doctors Visited	0
Age	0
Phyiscal Health	0
Mental Health	0
Dental Health	0
Employment	0
Stress Keeps Patient from Sleeping	0
Medication Keeps Patient from Sleeping	0
Pain Keeps Patient from Sleeping	0
Bathroom Needs Keeps Patient from Sleeping	0
Uknown Keeps Patient from Sleeping	0
Trouble Sleeping	0
Prescription Sleep Medication	0
Race	0
Gender	0

When I checked for duplicated values, I realized that I had 42 duplicated values.



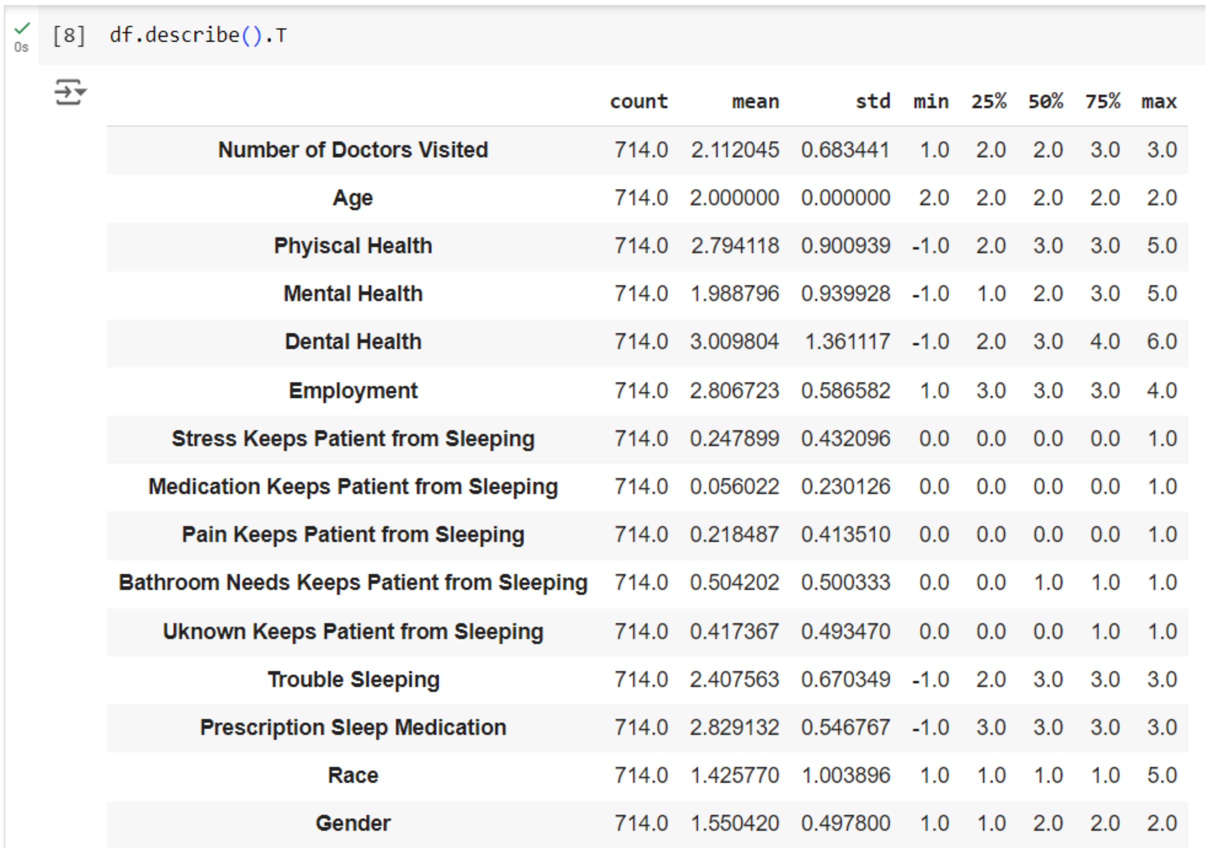
```
[10] df.duplicated().sum()

42
```

These were later dropped using the *df.drop_duplicates()* method which brought my dataset to have 672 rows and 15 columns. This changed from the original which was 714 rows.



```
#dropping or eliminating the duplicated values
df.drop_duplicates()
```

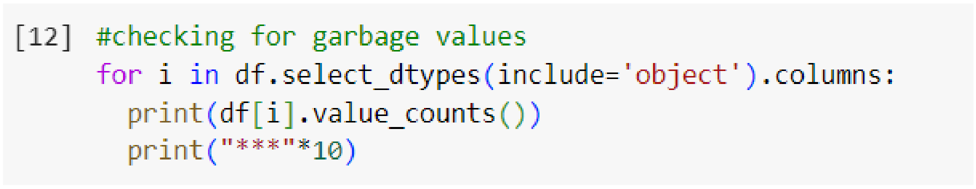
I also obtained a summary of my dataset using the pandas *describe()* method. The describe() function helped me to apply basic statistical computations on the dataset like extreme values, count of data points, and standard deviation, among others. This made me realize that any missing or NaN value is automatically skipped, giving a good picture of the distribution of data.



```
[8]  df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Number of Doctors Visited	714.0	2.112045	0.683441	1.0	2.0	2.0	3.0	3.0
Age	714.0	2.000000	0.000000	2.0	2.0	2.0	2.0	2.0
Phyiscal Health	714.0	2.794118	0.900939	-1.0	2.0	3.0	3.0	5.0
Mental Health	714.0	1.988796	0.939928	-1.0	1.0	2.0	3.0	5.0
Dental Health	714.0	3.009804	1.361117	-1.0	2.0	3.0	4.0	6.0
Employment	714.0	2.806723	0.586582	1.0	3.0	3.0	3.0	4.0
Stress Keeps Patient from Sleeping	714.0	0.247899	0.432096	0.0	0.0	0.0	0.0	1.0
Medication Keeps Patient from Sleeping	714.0	0.056022	0.230126	0.0	0.0	0.0	0.0	1.0
Pain Keeps Patient from Sleeping	714.0	0.218487	0.413510	0.0	0.0	0.0	0.0	1.0
Bathroom Needs Keeps Patient from Sleeping	714.0	0.504202	0.500333	0.0	0.0	1.0	1.0	1.0
Uknown Keeps Patient from Sleeping	714.0	0.417367	0.493470	0.0	0.0	0.0	1.0	1.0
Trouble Sleeping	714.0	2.407563	0.670349	-1.0	2.0	3.0	3.0	3.0
Prescription Sleep Medication	714.0	2.829132	0.546767	-1.0	3.0	3.0	3.0	3.0
Race	714.0	1.425770	1.003896	1.0	1.0	1.0	1.0	5.0
Gender	714.0	1.550420	0.497800	1.0	1.0	2.0	2.0	2.0

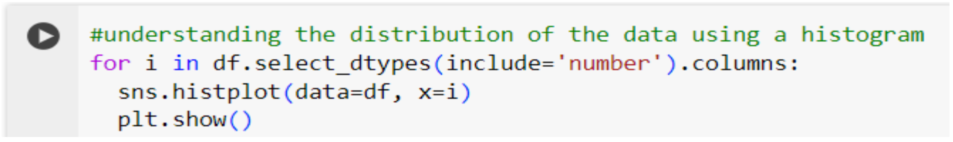
The garbage values (values whose data type is an object) were not identified in the dataset.



```
[12] #checking for garbage values
     for i in df.select_dtypes(include='object').columns:
       print(df[i].value_counts())
       print("***"*10)
```

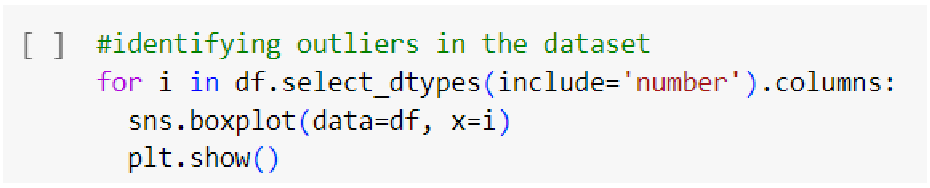
Visualizing data relationships;

During the process of visualizing data, I used a histogram to understand the distribution of data and it was well distributed.



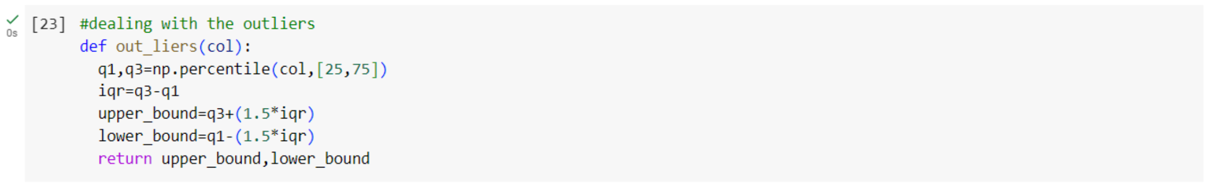
```
#understanding the distribution of the data using a histogram
for i in df.select_dtypes(include='number').columns:
  sns.histplot(data=df, x=i)
  plt.show()
```

The boxplot was also used to identify the outliers in my dataset. As a result of plotting the boxplot, I realized that my dataset had some outliers. The outliers were basically in physical health, employment, stress keeping patients from sleeping, medication keeping patients from sleeping, pain keeping patients from sleeping, trouble sleeping, prescription sleep medication, and race.

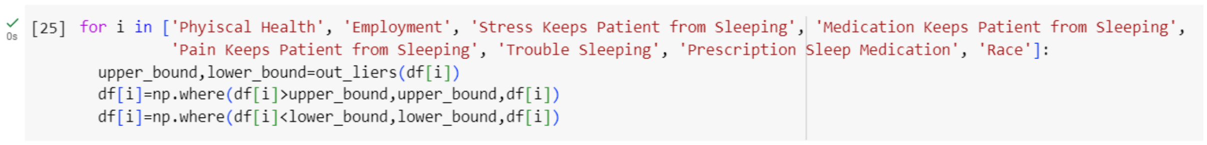


```
[ ] #identifying outliers in the dataset
    for i in df.select_dtypes(include='number').columns:
      sns.boxplot(data=df, x=i)
      plt.show()
```

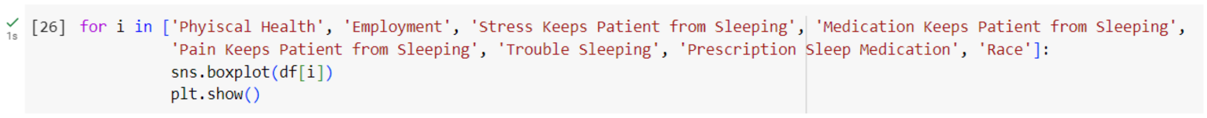
All the outliers in my dataset were dealt with using the 25th and 75th percentiles.



```
[23] #dealing with the outliers
     def out_liers(col):
       q1,q3=np.percentile(col,[25,75])
       iqr=q3-q1
       upper_bound=q3+(1.5*iqr)
       lower_bound=q1-(1.5*iqr)
       return upper_bound,lower_bound
```



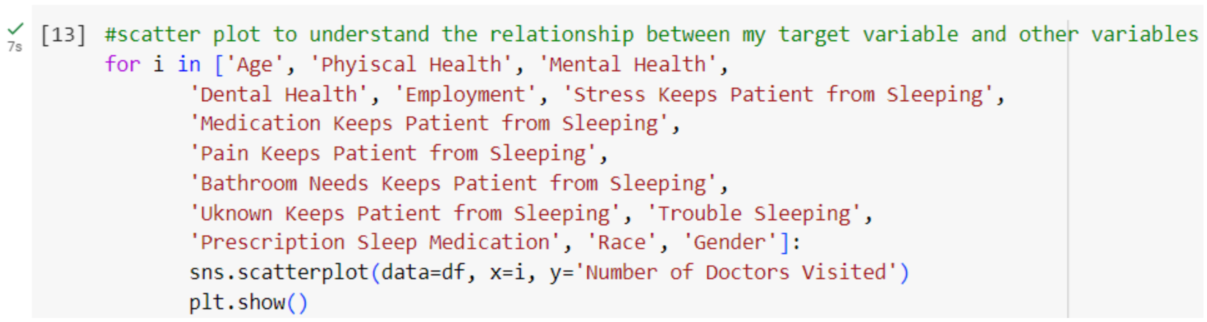
```
[25] for i in ['Phyiscal Health', 'Employment', 'Stress Keeps Patient from Sleeping', 'Medication Keeps Patient from Sleeping',
              'Pain Keeps Patient from Sleeping', 'Trouble Sleeping', 'Prescription Sleep Medication', 'Race']:
       upper_bound,lower_bound=out_liers(df[i])
       df[i]=np.where(df[i]>upper_bound,upper_bound,df[i])
       df[i]=np.where(df[i]<lower_bound,lower_bound,df[i])
```



```
[26] for i in ['Phyiscal Health', 'Employment', 'Stress Keeps Patient from Sleeping', 'Medication Keeps Patient from Sleeping',
              'Pain Keeps Patient from Sleeping', 'Trouble Sleeping', 'Prescription Sleep Medication', 'Race']:
              sns.boxplot(df[i])
              plt.show()
```

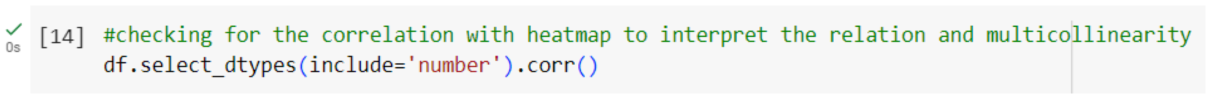
After the entire process, I checked for the outliers in the selected columns and I discovered that I never had outliers anymore. However, after dealing with the outliers, I discovered that it was affecting my data. Therefore I decided to leave them since they were not so much affecting my prediction.

In addition to the histogram and boxplot, I used the scatter plot as well to understand the relationship between my target variable "Number of doctors visited" and other variables. It was discovered that there is a positive relationship between my target variable and other variables in the dataset.

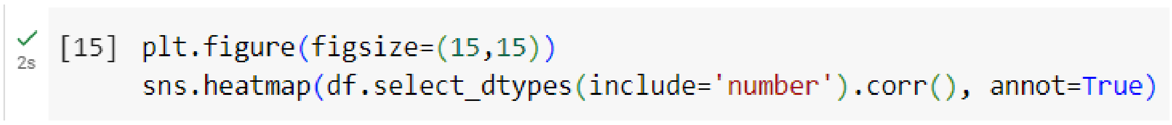


```
[13] #scatter plot to understand the relationship between my target variable and other variables
     for i in ['Age', 'Phyiscal Health', 'Mental Health',
             'Dental Health', 'Employment', 'Stress Keeps Patient from Sleeping',
             'Medication Keeps Patient from Sleeping',
             'Pain Keeps Patient from Sleeping',
             'Bathroom Needs Keeps Patient from Sleeping',
             'Uknown Keeps Patient from Sleeping', 'Trouble Sleeping',
             'Prescription Sleep Medication', 'Race', 'Gender']:
             sns.scatterplot(data=df, x=i, y='Number of Doctors Visited')
             plt.show()
```

Still visualizing data relationships, I used a heatmap chart to interpret the relationship and multicollinearity in my dataset.

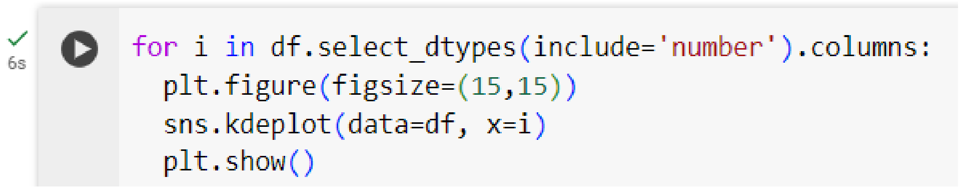


```
[14] #checking for the correlation with heatmap to interpret the relation and multicollinearity
     df.select_dtypes(include='number').corr()
```



```
[15] plt.figure(figsize=(15,15))
     sns.heatmap(df.select_dtypes(include='number').corr(), annot=True)
```

From the plot, it indicated the correlation between different variables, and the highest correlation was 1. Furthermore, I also explored the data using the kdeplot.



```
for i in df.select_dtypes(include='number').columns:
  plt.figure(figsize=(15,15))
  sns.kdeplot(data=df, x=i)
  plt.show()
```

Findings and insights about the dataset;

1. The dataset has no missing values and that makes it suitable to be used for the prediction.
2. The dataset has no duplicated values which makes it suitable for training the model.
3. From the visualization point of view, the data is well distributed and this makes it fit to be used to train the model for the prediction of the targeted variable.
4. The dataset has got few outliers which I have considered to be minor hence not affecting the final prediction after the model has been trained.
5. There is a positive correlation noticed between the target variable and other variables. This therefore makes the dataset fit to be used to train the model to arrive at the final prediction.
6. I have met a barrier of fixing the outliers where by considering the 25th and 75th percentile, most of the outliers in the dataset tend to be close to the 25th percentile and once used to remove the outliers it somehow affects my dataset. Therefore, it is based on this fact that I have chosen to first comment out the code for fixing the outliers and first observe how it works.

I, therefore, conclude that with the dataset that I have chosen to use to train the model and my target variable together with other variables, it is so interesting to discover how someone can age or grow old when he or she is still healthy.