SLEEPING DISORDER PREDICTION

In [7]:
```
import pandas as pd
```

In [8]:
```
data=pd.read_csv('sleepdisorder.csv')
```

DISPLAYING TOP 5 ROWS OF DATA SET

In [9]:
```
data.head()
```

Out[9]:

	gender	age	weight	occupation	headache	myalgia	EEG	haemoglobin	b12 deficiency	stress	trauma	depressi
0	male	61	70	watchman	yes	yes	9	9.2	no	yes	no	y
1	male	23	63	student	yes	yes	6	8.7	no	no	no	
2	female	32	43	housewife	no	no	5	6.2	no	no	no	
3	female	24	47	teacher	no	no	8	11.2	no	no	no	
4	male	72	72	retired	yes	no	10	6.2	yes	no	no	

DISPLAY LAST 5 ROWS OF DATA SET

In [10]:
```
data.tail()
```

Out[10]:

	gender	age	weight	occupation	headache	myalgia	EEG	haemoglobin	b12 deficiency	stress	trauma	depres
91	femalle	42	180	housewife	yes	yes	10	8.0	yes	no	no	
92	male	29	140	IT	yes	yes	5	11.0	yes	yes	yes	
93	female	38	40	housewife	yes	yes	10	6.0	yes	yes	no	
94	male	52	60	teacher	no	yes	10	11.0	no	no	no	
95	female	17	40	student	yes	yes	5	11.0	no	no	no	

INFORMATION ABOUT THE DATA SET

In [11]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96 entries, 0 to 95
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             96 non-null     object 
 1   age                96 non-null     int64  
 2   weight             96 non-null     int64  
 3   occupation         96 non-null     object 
 4   headache           96 non-null     object 
 5   myalgia            94 non-null     object 
 6   EEG                96 non-null     int64  
 7   haemoglobin        96 non-null     float64
 8   b12 deficiency     96 non-null     object 
 9   stress             95 non-null     object 
 10  trauma             95 non-null     object 
 11  depression         96 non-null     object 
 12  drugs              96 non-null     object
 13  B.P                95 non-null     float64
 14  vitamin e          96 non-null     object
 15  vitamin d          96 non-null     object
 16  sleeping disorder  96 non-null     object
dtypes: float64(2), int64(3), object(12)
memory usage: 12.9+ KB
```

CHECK FOR NULL VALUES

In [12]:

```python
data.isnull().sum()
```

Out[12]:

```
gender               0
age                  0
weight               0
occupation           0
headache             0
myalgia              2
EEG                  0
haemoglobin          0
b12 deficiency       0
stress               1
trauma               1
depression           0
drugs                0
B.P                  1
vitamin e            0
vitamin d            0
sleeping disorder    0
dtype: int64
```

REMOVING FEATURES THAT ARE NOT IMPORTANT IN PREDICTING SLEEPING DISORDER

In [13]:

```python
data.drop(['gender','age'],axis=1,inplace=True)
```

In [14]:

```python
data.head()
```

Out[14]:

	weight	occupation	headache	myalgia	EEG	haemoglobin	b12 deficiency	stress	trauma	depression	drugs	B
0	70	watchman	yes	yes	9	9.2	no	yes	no	yes	yes	68
1	63	student	yes	yes	6	8.7	no	no	no	no	yes	72
2	43	housewife	no	no	5	6.2	no	no	no	no	no	80
3	47	teacher	no	no	8	11.2	no	no	no	no	yes	68
4	72	retired	yes	no	10	6.2	yes	no	no	no	yes	58

ENCODING THE CATEGORICAL FEATURES

In [15]:

```python
from sklearn.preprocessing import LabelEncoder
```

In [16]:

```python
n_weight=LabelEncoder()
n_occupation=LabelEncoder()
n_headache=LabelEncoder()
n_myalgia=LabelEncoder()
n_b12_deficiency=LabelEncoder()
n_stress=LabelEncoder()
n_trauma=LabelEncoder()
n_depression=LabelEncoder()
n_drugs=LabelEncoder()
```

```
n_vitamin_E=LabelEncoder()
n_vitamin_D=LabelEncoder()
n_sleeping_disorder=LabelEncoder()
```

In [17]:

```
data['weight_']=n_weight.fit_transform(data['weight'])
data['occupation_']=n_weight.fit_transform(data['occupation'])
data['headach_']=n_weight.fit_transform(data['headache'])
data['myalgia_']=n_weight.fit_transform(data['myalgia'])
data['b12_deficiency']=n_b12_deficiency.fit_transform(data['b12 deficiency'])
data['stress_']=n_stress.fit_transform(data['stress'])
data['trauma_']=n_trauma.fit_transform(data['trauma'])
data['depression_']=n_depression.fit_transform(data['depression'])
data['drugs_']=n_drugs.fit_transform(data['drugs'])
data['vitamin_e']=n_vitamin_E.fit_transform(data['vitamin e '])
data['vitamin_d']=n_vitamin_D.fit_transform(data['vitamin d'])
data['sleeping_disorder']=n_sleeping_disorder.fit_transform(data['sleeping disorder'])
```

In [18]:

```
data.head()
```

Out[18]:

	weight	occupation	headache	myalgia	EEG	haemoglobin	b12 deficiency	stress	trauma	depression	...	heada
0	70	watchman	yes	yes	9	9.2	no	yes	no	yes	...	
1	63	student	yes	yes	6	8.7	no	no	no	no	...	
2	43	housewife	no	no	5	6.2	no	no	no	no	...	
3	47	teacher	no	no	8	11.2	no	no	no	no	...	
4	72	retired	yes	no	10	6.2	yes	no	no	no	...	

5 rows × 27 columns

In [19]:

```
data.drop(['weight','occupation','headache','myalgia','b12 deficiency','stress','trauma'
```

In [20]:

```
data.head()
```

Out[20]:

	EEG	haemoglobin	B.P	weight_	occupation_	headach_	myalgia_	b12_deficiency	stress_	trauma_	depressi
0	9	9.2	68.0	8	6	1	1	0	1	0	
1	6	8.7	72.0	7	4	1	1	0	0	0	
2	5	6.2	80.0	2	2	0	0	0	0	0	
3	8	11.2	68.0	3	5	0	0	0	0	0	
4	10	6.2	58.0	9	3	1	0	1	0	0	

CHECKING NULL VALUES AND REMOVING NULL VALUES

In [21]:

```
data.isnull().sum()
```

Out[21]:

```
EEG                 0
haemoglobin         0
B.P                 1
weight_             0
occupation_         0
headach_            0
myalgia_            0
b12_deficiency      0
stress_             0
trauma_             0
```

```
depression_         0
drugs_              0
vitamin_e           0
vitamin_d           0
sleeping_disorder   0
dtype: int64
```

In [22]:
```
data['B.P']=data['B.P'].interpolate()
```

In [23]:
```
data.isnull().sum()  # we have no null values
```

Out[23]:
```
EEG                 0
haemoglobin         0
B.P                 0
weight_             0
occupation_         0
headach_            0
myalgia_            0
b12_deficiency      0
stress_             0
trauma_             0
depression_         0
drugs_              0
vitamin_e           0
vitamin_d           0
sleeping_disorder   0
dtype: int64
```

SELECTING INPUT AND TARGET FEATURES

In [24]:
```
x=data.drop('sleeping_disorder',axis=1)
y=data['sleeping_disorder']
```

SPLITTING TRAINING DATA AND TEST DATA

In [25]:
```
from sklearn.model_selection import train_test_split
```

In [26]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=21)
```

SCIKIT-LEARN PIPELINE

In [27]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.pipeline import Pipeline
```

In [28]:
```
pipeline_lr=Pipeline([('scalar1',StandardScaler()),
                     ('lr_classifer',LogisticRegression())])
pipeline_knn=Pipeline([('scalar2',StandardScaler()),
                      ('knn_classifier',KNeighborsClassifier())])
pipeline_svc=Pipeline([('scalar3',StandardScaler()),
                      ('svc_classifier',SVC())])
pipeline_nb=Pipeline([('scalar4',StandardScaler()),
                     ('nb_classifier',GaussianNB())])
pipeline_dt=Pipeline([('dt_classifier',DecisionTreeClassifier())])
pipeline_rf = Pipeline([('rf_classifier',RandomForestClassifier(max_depth=3))])
```

In [29]:
```
pipelines=[pipeline_lr,pipeline_knn,pipeline_svc,pipeline_nb,pipeline_dt,pipeline_rf]
```

In [30]:
```
pipelines
```

Out[30]:
```
[Pipeline(steps=[('scalar1', StandardScaler()),
                 ('lr_classifer', LogisticRegression())]),
 Pipeline(steps=[('scalar2', StandardScaler()),
                 ('knn_classifier', KNeighborsClassifier())]),
 Pipeline(steps=[('scalar3', StandardScaler()), ('svc_classifier', SVC())]),
 Pipeline(steps=[('scalar4', StandardScaler()), ('nb_classifier', GaussianNB())]),
 Pipeline(steps=[('dt_classifier', DecisionTreeClassifier())]),
 Pipeline(steps=[('rf_classifier', RandomForestClassifier(max_depth=3))])]
```

In [31]:
```
for pipe in pipelines:
    pipe.fit(x_train,y_train)
```

In [32]:
```
pipe_dict={
    0:'Logistic Regression',
    1:'KNeighborsClassifier',
    2:'SupportVectorMachine',
    3:'NaiveBayes',
    4:'DecisionTree',
    5:'RandomForest'
}
```

In [33]:
```
pipe_dict
```

Out[33]:
```
{0: 'Logistic Regression',
 1: 'KNeighborsClassifier',
 2: 'SupportVectorMachine',
 3: 'NaiveBayes',
 4: 'DecisionTree',
 5: 'RandomForest'}
```

In [34]:
```
for i,model in enumerate(pipelines):
    print('{} test accuracy is {}'.format(pipe_dict[i],model.score(x_test,y_test)*100))
```

```
Logistic Regression test accuracy is 80.0
KNeighborsClassifier test accuracy is 65.0
SupportVectorMachine test accuracy is 80.0
NaiveBayes test accuracy is 70.0
DecisionTree test accuracy is 85.0
RandomForest test accuracy is 75.0
```

```
C:\Users\Rohith\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: Fu
tureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default beh
avior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this behavi
or will change: the default value of `keepdims` will become False, the `axis` over which
the statistic is taken will be eliminated, and the value None will no longer be accepte
d. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

so, we found that decision tree gives us the highest accuracy

In [35]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [36]:
```
x=data.drop('sleeping_disorder',axis=1)
y=data['sleeping_disorder']
x
```

Out[36]:

	EEG	haemoglobin	B.P	weight_	occupation_	headach_	myalgia_	b12_deficiency	stress_	trauma_	depre
0	9	9.2	68.0	8	6	1	1	0	1	0	
1	6	8.7	72.0	7	4	1	1	0	0	0	
2	5	6.2	80.0	2	2	0	0	0	0	0	

3	8	11.2	68.0	3	5	0	0	0	0	0
4	10	6.2	58.0	9	3	1	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...
91	10	8.0	130.0	15	2	1	1	1	0	0
92	5	11.0	120.0	14	0	1	1	1	1	1
93	10	6.0	140.0	1	2	1	1	1	1	0
94	10	11.0	60.0	6	5	0	1	0	0	0
95	5	11.0	50.0	1	4	1	1	0	0	0

96 rows × 14 columns

In [37]:

```
obj=DecisionTreeClassifier()
obj.fit(x,y)
```

Out[37]:

```
DecisionTreeClassifier()
```

SAVING THE MODEL USING JOBLIB

In [38]:

```
import joblib
```

In [39]:

```
joblib.dump(obj,'model_sleeping')
```

Out[39]:

```
['model_sleeping']
```

GUI

In [40]:

```
from tkinter import *
import joblib

import numpy as np
from sklearn import *
def show_entry_fields():
    p1=float(e1.get())
    p2=float(e2.get())
    p3=float(e3.get())
    p4=float(e4.get())
    p5=float(e5.get())
    p6=float(e6.get())
    p7=float(e7.get())
    p8=float(e8.get())
    p9=float(e9.get())
    p10=float(e10.get())
    p11=float(e11.get())
    p12=float(e12.get())
    p13=float(e13.get())
    p14=float(e14.get())




    model = joblib.load('model_sleeping')
    result=model.predict([[p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14]])

    if result == 0:
        Label(master, text="patient does not have sleeping disorder").grid(row=50)
    else:
        Label(master, text="patient has sleeping disorder").grid(row=50)


master = Tk()
master.title("Sleeping disorder prediction using machine learning")


label = Label(master, text = "Sleeping disorder Prediction Using Machine Learning"
                          , bg = "black", fg = "white"). \
                               grid(row=0,columnspan=2)


Label(master, text="Enter EEG").grid(row=1)
Label(master, text="Enter hemoglobin").grid(row=2)
Label(master, text="Enter B.P").grid(row=3)
Label(master, text="Enter Weight ").grid(row=4)
Label(master, text="Enter Occupation ").grid(row=5)
Label(master, text="Enter headache status").grid(row=6)
Label(master, text="Enter Myalgia status").grid(row=7)
Label(master, text="B12 deficiency? ").grid(row=8)
Label(master, text="experiencing stress? ").grid(row=9)
Label(master, text="experiencing trauma? ").grid(row=10)
Label(master, text="experiencing Depression? ").grid(row=11)
Label(master, text="Drug intake ").grid(row=12)
Label(master, text="Vitamin-D deficiency? ").grid(row=13)
Label(master, text="Vitamin-E deficiency? ").grid(row=14)


e1 = Entry(master)
e2 = Entry(master)
e3 = Entry(master)
e4 = Entry(master)
e5 = Entry(master)
e6 = Entry(master)
e7 = Entry(master)
e8 = Entry(master)
e9 = Entry(master)
e10 = Entry(master)
e11 = Entry(master)
e12 = Entry(master)
e13 = Entry(master)
e14 = Entry(master)



e1.grid(row=1, column=1)
e2.grid(row=2, column=1)
e3.grid(row=3, column=1)
e4.grid(row=4, column=1)
e5.grid(row=5, column=1)
e6.grid(row=6, column=1)
e7.grid(row=7, column=1)
e8.grid(row=8, column=1)
e9.grid(row=9, column=1)
e10.grid(row=10, column=1)
e11.grid(row=11, column=1)
e12.grid(row=12, column=1)
e13.grid(row=13, column=1)
e14.grid(row=14, column=1)





Button(master, text='Predict', command=show_entry_fields).grid()
```

```
mainloop()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: