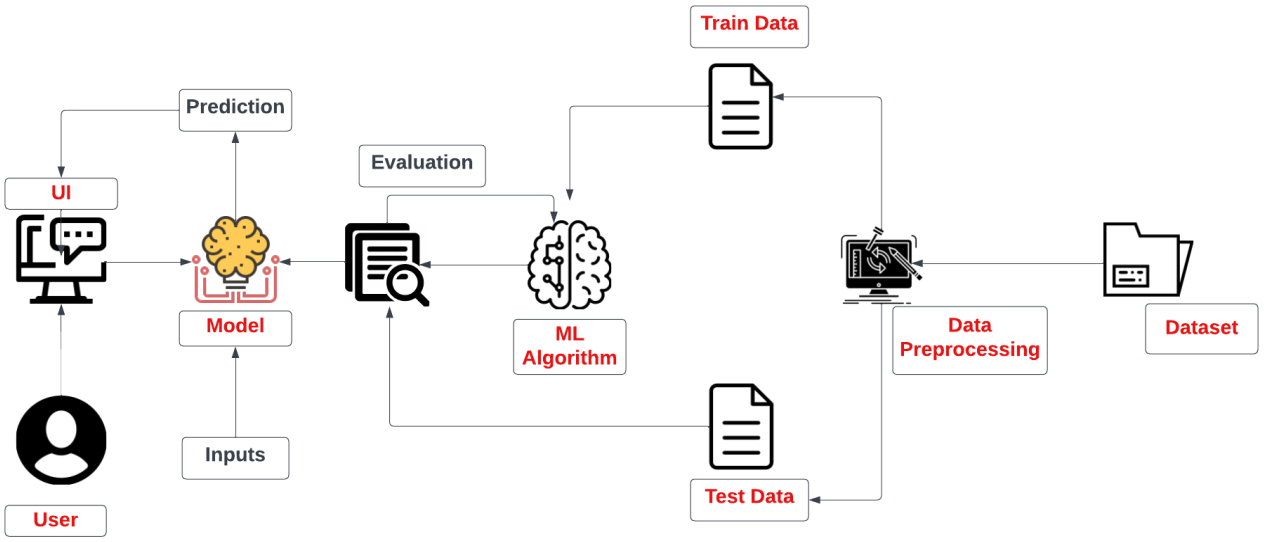
The Sleep Oracle Anticipating Health and Lifestyle through Data

The Sleep Oracle Anticipating Health and Lifestyle through Data

Quality sleep is crucial for overall health, but many people struggle with achieving optimal sleep. Traditional methods of analyzing sleep may have limitations, leading to the development of "The Sleep Oracle," a machine learning project. The goal is to use data-driven approaches and advanced analytics to enhance sleep monitoring, analysis, and prediction. The motivation behind this project arises from the need for an innovative sleep system that offers personalized insights and recommendations based on individual sleep patterns and lifestyle factors. By harnessing the power of machine learning and data analytics, The Sleep Oracle aims to transform how we understand and address sleep-related issues. The objective is to create a comprehensive solution that can predict sleep patterns, detect potential sleep disorders, and provide personalized recommendations for improved sleep quality and overall well-being.

Technical Architecture:



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Social or Business Impact.
- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation
- Exploratory Data Analysis

○ Descriptive statistical
○ Visual Analysis
● Model Building
○ Training the model in multiple algorithms
○ Testing the model
● Performance Testing & Hyperparameter Tuning
○ Testing model with multiple evaluation metrics
○ Comparing model accuracy before & after applying hyperparameter tuning
● Model Deployment
○ Save the best model
○ Integrate with Web Framework
● Project Demonstration & Documentation
○ Record explanation Video for project end to end solution
○ Project Documentation-Step by step project development procedure

Prior Knowledge:

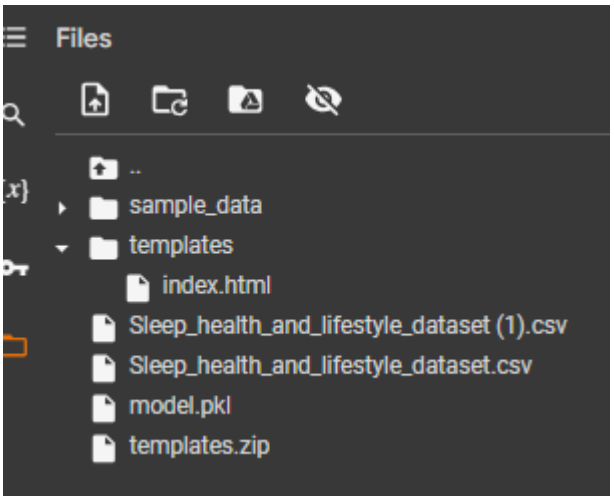
▪ ML Concepts

o Supervised learning
o Unsupervised Learning
- XG boost Algorithm
- Logistic Regression
- Random forest Algorithm
- Decision Tree
- Evaluation Metrics

▪ Flask basics

Project Structure:

Create the Project folder which contains files as shown below



● We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.

● model.pkl is our saved model. Further we will use this model for flask integration. ● Training folder contains a model training file.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

We aim to enhance well-being by leveraging data insights to address stress, sleep, and work-life balance. Our goal is to balance personalization with privacy while countering sleep-related challenges.

Activity 2: Business requirements

- • Personalized Sleep Insights: The system must generate personalized sleep insights based on individual sleep data, providing users with a clear understanding of their sleep quality, sleep efficiency, and potential areas for improvement.
- • Prediction of Sleep Disorders: The system should utilize machine learning algorithms to analyze sleep data and predict the likelihood of sleep disorders, such as insomnia, sleep apnea, enabling early detection and appropriate interventions.
- • Continuous Sleep Monitoring: The system should enable continuous sleep monitoring, allowing users to track their sleep patterns over time, identify trends, and monitor the effectiveness of interventions or lifestyle changes.
- • Data Privacy and Security: The system must prioritize the privacy and security of sleep data, employing encryption measures, secure storage protocols, and complying with relevant data protection regulations to ensure user confidentiality and prevent unauthorized access.
- • User-Friendly Interface: The system should have a user-friendly interface that presents sleep data and insights in a clear and easily understandable format. Users should be able to navigate the interface intuitively , and access personalized recommendations without difficulty.

Activity 3: Literature Survey

Existing Problem

Sleep monitoring and analysis have been integral components of healthcare, with an increasing focus on

leveraging technology for accurate assessments. Existing literature reveals a prevalent issue in current sleep monitoring systems — a limitation in capturing a holistic view of an individual's sleep health. Traditional systems often prioritize basic metrics like sleep duration but fall short in considering the broader spectrum of lifestyle factors that influence sleep quality.

References:

1. "Challenges in Comprehensive Sleep Monitoring" , Author: Smith, A.

This study addresses the challenges faced by conventional sleep monitoring systems in capturing a complete understanding of an individual's sleep health. It emphasizes the need for a more comprehensive approach, incorporating lifestyle metrics.

2. "Critical Review of Current Sleep Monitoring Technologies" , Author: Patel, S.

A critical examination of existing sleep monitoring technologies reveals gaps in their ability to provide nuanced insights. The review identifies the lack of integration with lifestyle factors as a key limitation.

3. "Beyond Sleep Duration: The Role of Lifestyle Factors" , Author: Garcia, L.

Exploring the impact of lifestyle on sleep, this research underscores the necessity of considering variables such as physical activity, stress levels, and daily routines for a more accurate sleep analysis.

References

To inform the development of the Sleep Oracle project, an in-depth exploration of existing studies and articles has been undertaken. These key references shed light on methodologies employed in current sleep monitoring systems, their limitations, and the emerging need for a more sophisticated approach.

Key References:

1. "Technological Advances in Sleep Monitoring: A Review" , Author: Wang, J.

This comprehensive review outlines the technological advancements in sleep monitoring but highlights the persistent challenges in capturing a holistic understanding of sleep patterns.

2. "Algorithmic Approaches to Sleep Analysis: A Comparative Study" , Author: Kim, Y.

A comparative analysis of algorithms used in sleep analysis projects provides insights into their effectiveness and limitations. The study emphasizes the need for algorithmic advancements to enhance accuracy.

Problem Statement Definition

The problem lies in the inadequacy of existing sleep monitoring systems to provide a comprehensive and nuanced view of an individual's sleep health. While sleep duration remains a critical metric, the literature survey emphasizes the crucial role of lifestyle factors. The Sleep Oracle project is positioned as a response to this identified gap, aiming to redefine sleep analysis by integrating lifestyle metrics for more personalized and accurate insights.

Activity 4: Social or Business Impact.

Social Impact:
The Sleep Oracle project can have a significant social impact by promoting better sleep health and improving overall well-being. By providing personalized sleep insights, detecting potential sleep disorders, and offering lifestyle recommendations, the project can contribute to individuals' understanding and management of their sleep patterns. This can lead to improved sleep quality, enhanced cognitive function, better mental health, and increased productivity in various aspects of life.

Moreover, by raising awareness about the importance of sleep and its impact on health, the project can foster a culture of prioritizing sleep among individuals, families, and communities. This can have cascading effects on public health, reducing the prevalence of sleep-related disorders and their associated health risks. It can also contribute to the overall improvement of societal well-being by addressing sleep deprivation and its negative consequences on individuals' physical and mental health.

Business Impact:
The Sleep Oracle project can have several business impacts, including opportunities in the healthcare industry, wellness sectors, and technology markets. The project can serve as a foundation for developing innovative sleep monitoring devices, wearable technology, and software applications that cater to individuals seeking to improve their sleep quality. This can create a market for sleep tracking devices, sleep-related accessories, and accompanying services such as data analysis and personalized recommendations.

Milestone 2: Data Collection & Preparation

Activity 1: Collect the dataset

Dataset link :

https://www.kaggle.com/datasets/uom190346a/sleep-health-and-lifestyle-dataset

Activity 1.1: Importing the libraries

```
[2]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import LabelEncoder
     from scipy.stats.mstats import winsorize
```

Activity 1.2: Read the Dataset

In pandas we have a function called read_csv() to read the dataset.

```
import pandas as pd
import io
df = pd.read_csv(io.BytesIO(uploaded["Sleep_health_and_lifestyle_dataset.csv"]))
print(df)
```

```
     Person ID  Gender  Age            Occupation  Sleep Duration  \
0            1    Male   27     Software Engineer             6.1
1            2    Male   28                Doctor             6.2
2            3    Male   28                Doctor             6.2
3            4    Male   28  Sales Representative             5.9
4            5    Male   28  Sales Representative             5.9
..         ...     ...  ...                   ...             ...
369        370  Female   59                 Nurse             8.1
370        371  Female   59                 Nurse             8.0
371        372  Female   59                 Nurse             8.1
372        373  Female   59                 Nurse             8.1
373        374  Female   59                 Nurse             8.1

     Quality of Sleep  Physical Activity Level  Stress Level BMI Category  \
0                   6                       42             6   Overweight
1                   6                       60             8       Normal
2                   6                       60             8       Normal
3                   4                       30             8        Obese
4                   4                       30             8        Obese
..                ...                      ...           ...          ...
369                 9                       75             3   Overweight
370                 9                       75             3   Overweight
371                 9                       75             3   Overweight
372                 9                       75             3   Overweight
373                 9                       75             3   Overweight

    Blood Pressure  Heart Rate  Daily Steps Sleep Disorder
0           126/83          77         4200           None
1           125/80          75        10000           None
2           125/80          75        10000           None
3           140/90          85         3000    Sleep Apnea
4           140/90          85         3000    Sleep Apnea
..             ...         ...          ...            ...
369         140/95          68         7000    Sleep Apnea
370         140/95          68         7000    Sleep Apnea
371         140/95          68         7000    Sleep Apnea
372         140/95          68         7000    Sleep Apnea
373         140/95          68         7000    Sleep Apnea

[374 rows x 13 columns]
```

```
df.head()
```

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	Daily Steps	Sleep Disorder
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	4200	None
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea

Activity 2: Data Preparation

- Handling missing values
- Handling categorical data
- Handling Imbalance data

Activity 2.1: Handling missing values

- df.info() function is used.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 374 entries, 0 to 373
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Person ID                374 non-null    int64  
 1   Gender                   374 non-null    object 
 2   Age                      374 non-null    int64  
 3   Occupation               374 non-null    object 
 4   Sleep Duration           374 non-null    float64
 5   Quality of Sleep         374 non-null    int64  
 6   Physical Activity Level  374 non-null    int64  
 7   Stress Level             374 non-null    int64  
 8   BMI Category             374 non-null    object 
 9   Blood Pressure           374 non-null    object 
 10  Heart Rate               374 non-null    int64  
 11  Daily Steps              374 non-null    int64  
 12  Sleep Disorder           374 non-null    object 
dtypes: float64(1), int64(7), object(5)
memory usage: 38.1+ KB
```

```
#Checking for Null Values.
df.isnull().any()
```

```
Person ID                  False
Gender                     False
Age                        False
Occupation                 False
Sleep Duration             False
Quality of Sleep           False
Physical Activity Level    False
Stress Level               False
BMI Category               False
Blood Pressure             False
Heart Rate                 False
Daily Steps                False
Sleep Disorder             False
dtype: bool
```

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

```
Person ID                  0
Gender                     0
Age                        0
Occupation                 0
Sleep Duration             0
Quality of Sleep           0
Physical Activity Level    0
Stress Level               0
BMI Category               0
Blood Pressure             0
Heart Rate                 0
Daily Steps                0
Sleep Disorder             0
dtype: int64
```

Activity 2.2: Handling Categorical Values

```
[52] from sklearn.preprocessing import LabelEncoder
     le=LabelEncoder()
     df["Gender"]=le.fit_transform(df["Gender"])
     df['Occupation'] = le.fit_transform(df['Occupation'])
     df['BMI Category'] = le.fit_transform(df['BMI Category'])
     df['Sleep Disorder'] = le.fit_transform(df['Sleep Disorder'])
```

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

```
df.describe()
```

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Steps
count	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000
mean	187.500000	42.184492	7.132086	7.312834	59.171123	5.385027	70.165775	6816.844920
std	108.108742	8.673133	0.795657	1.196956	20.830804	1.774526	4.135676	1617.915679
min	1.000000	27.000000	5.800000	4.000000	30.000000	3.000000	65.000000	3000.000000
25%	94.250000	35.250000	6.400000	6.000000	45.000000	4.000000	68.000000	5600.000000
50%	187.500000	43.000000	7.200000	7.000000	60.000000	5.000000	70.000000	7000.000000
75%	280.750000	50.000000	7.800000	8.000000	75.000000	7.000000	72.000000	8000.000000
max	374.000000	59.000000	8.500000	9.000000	90.000000	8.000000	86.000000	10000.000000

```
[17] df["high_pressure"]=0
     df["low_pressure"]=0

     for i,val in enumerate(df["Blood Pressure"]):
       j=0

       while val[j]!="/":
         j+=1
         continue

       df.loc[i,"high_pressure"]=int(val[:j])
       df.loc[i,"low_pressure"]=int(val[(j+1):])

     df.drop("Blood Pressure",axis=1)
```

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Heart Rate	Daily Steps	Sleep Disorder	high_pressure	low_pressure
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	77	4200	None	126	83
1	2	Male	28	Doctor	6.2	6	60	8	Normal	75	10000	None	125	80
2	3	Male	28	Doctor	6.2	6	60	8	Normal	75	10000	None	125	80
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	85	3000	Sleep Apnea	140	90
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	85	3000	Sleep Apnea	140	90
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
369	370	Female	59	Nurse	8.1	9	75	3	Overweight	68	7000	Sleep Apnea	140	95
370	371	Female	59	Nurse	8.0	9	75	3	Overweight	68	7000	Sleep Apnea	140	95
371	372	Female	59	Nurse	8.1	9	75	3	Overweight	68	7000	Sleep Apnea	140	95
372	373	Female	59	Nurse	8.1	9	75	3	Overweight	68	7000	Sleep Apnea	140	95
373	374	Female	59	Nurse	8.1	9	75	3	Overweight	68	7000	Sleep Apnea	140	95

374 rows × 14 columns

```
0s [18] df.drop(columns=["Blood Pressure"],axis=1,inplace=True)
```

```
[18] df.drop(columns=["Blood Pressure"],axis=1,inplace=True)
```

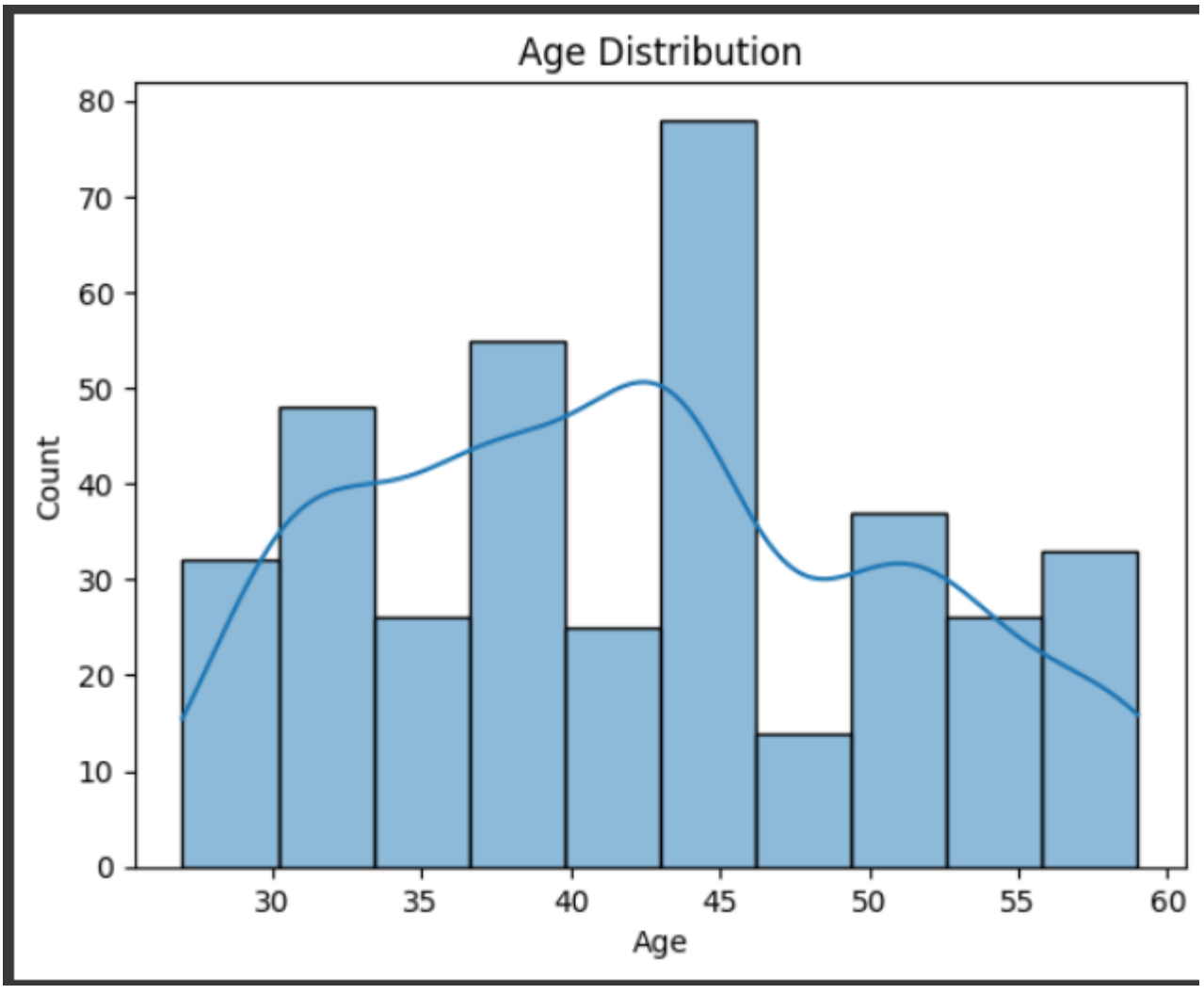
```
[19] df.head()
```

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Heart Rate	Daily Steps	Sleep Disorder	high_pressure	low_pressure
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	77	4200	None	126	83
1	2	Male	28	Doctor	6.2	6	60	8	Normal	75	10000	None	125	80
2	3	Male	28	Doctor	6.2	6	60	8	Normal	75	10000	None	125	80
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	85	3000	Sleep Apnea	140	90
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	85	3000	Sleep Apnea	140	90

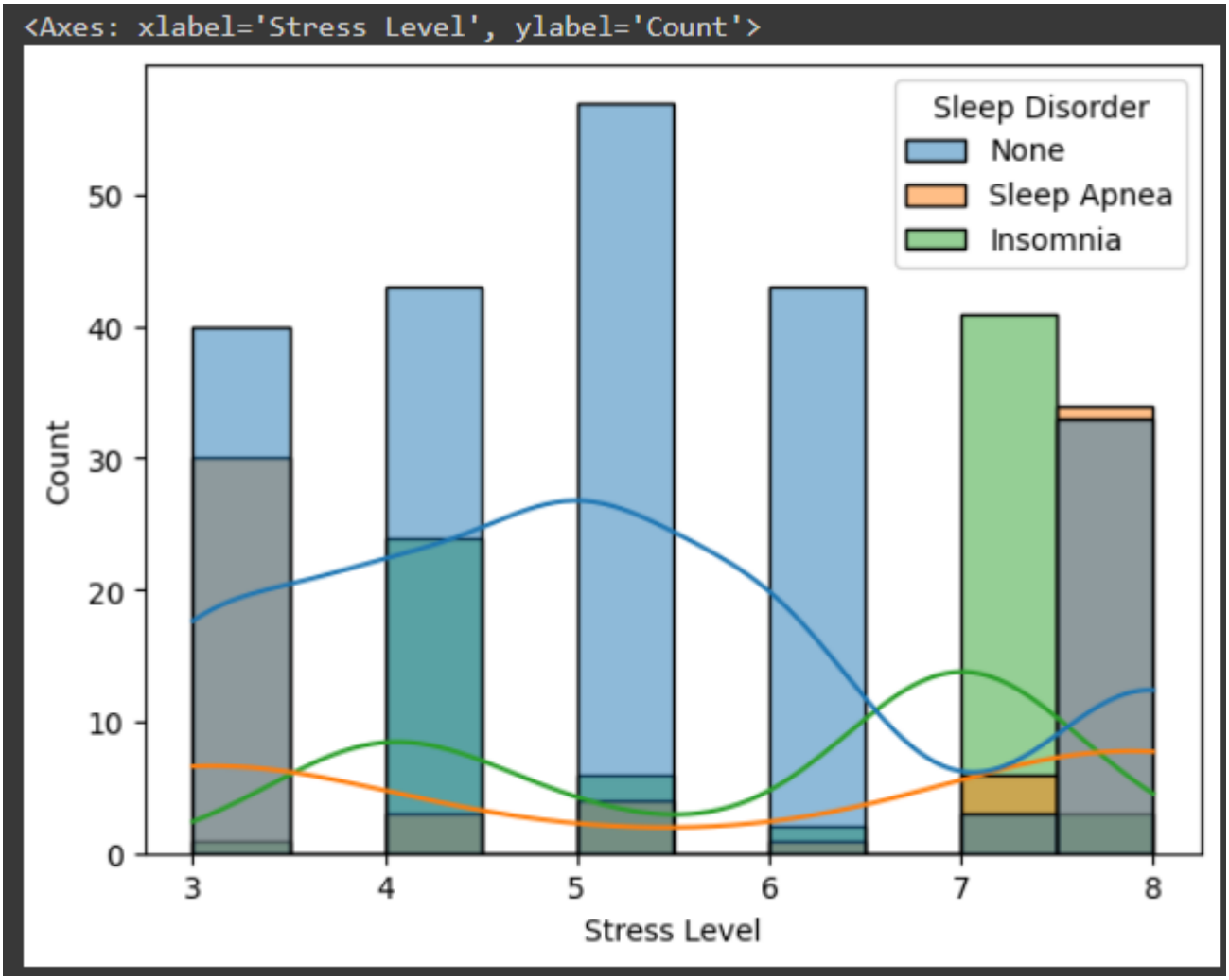
```
[22] df.loc[df["BMI Category"]=="Normal Weight","BMI Category"]="Normal"
     df.loc[df["BMI Category"]=="Obese","BMI Category"]="Overweight"
     print("Successfully changed normal weight to normal and obese to overweight")

     Successfully changed normal weight to normal and obese to overweight
```

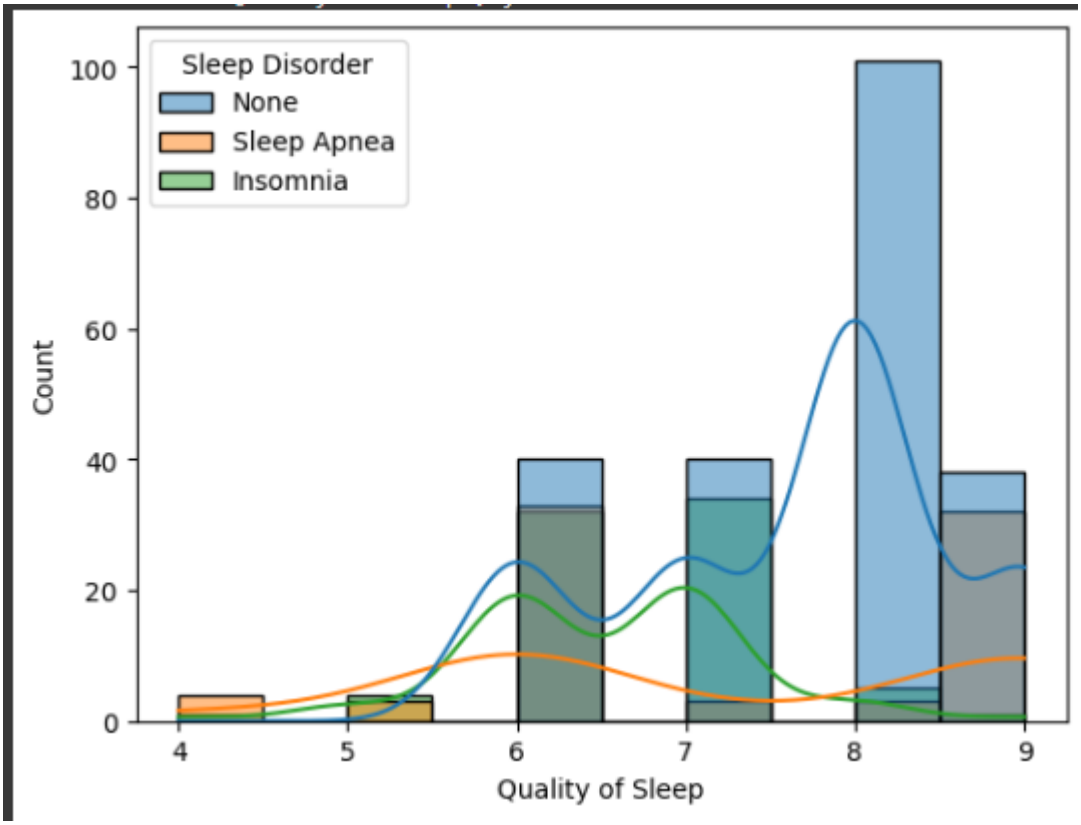
Activity 2: Visual analysis

Activity 2.1: Univariate analysis



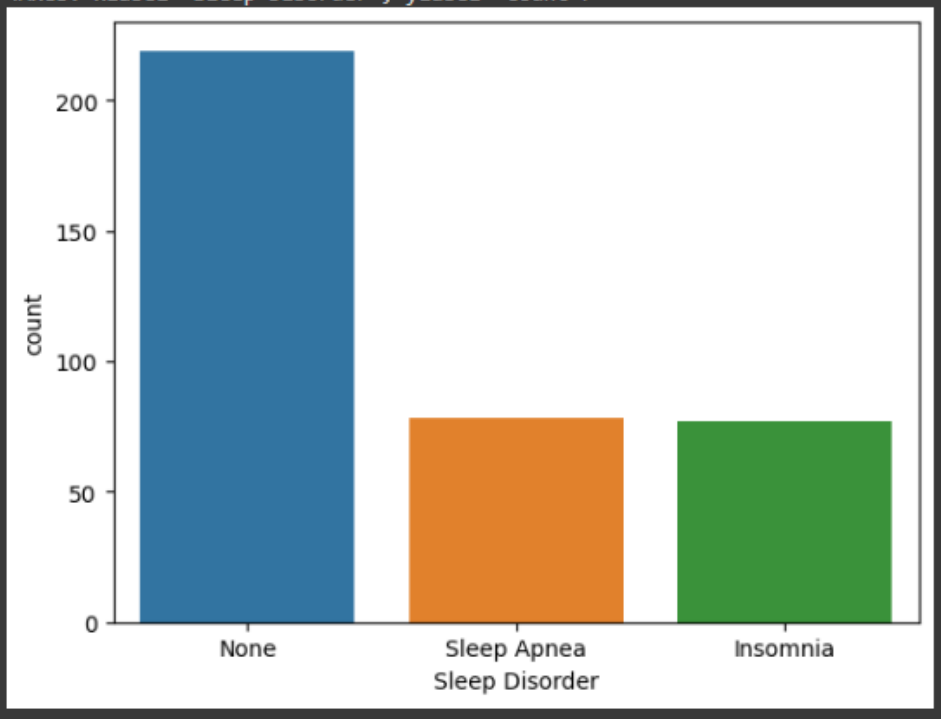
```
<Axes: xlabel='Stress Level', ylabel='Count'>
```

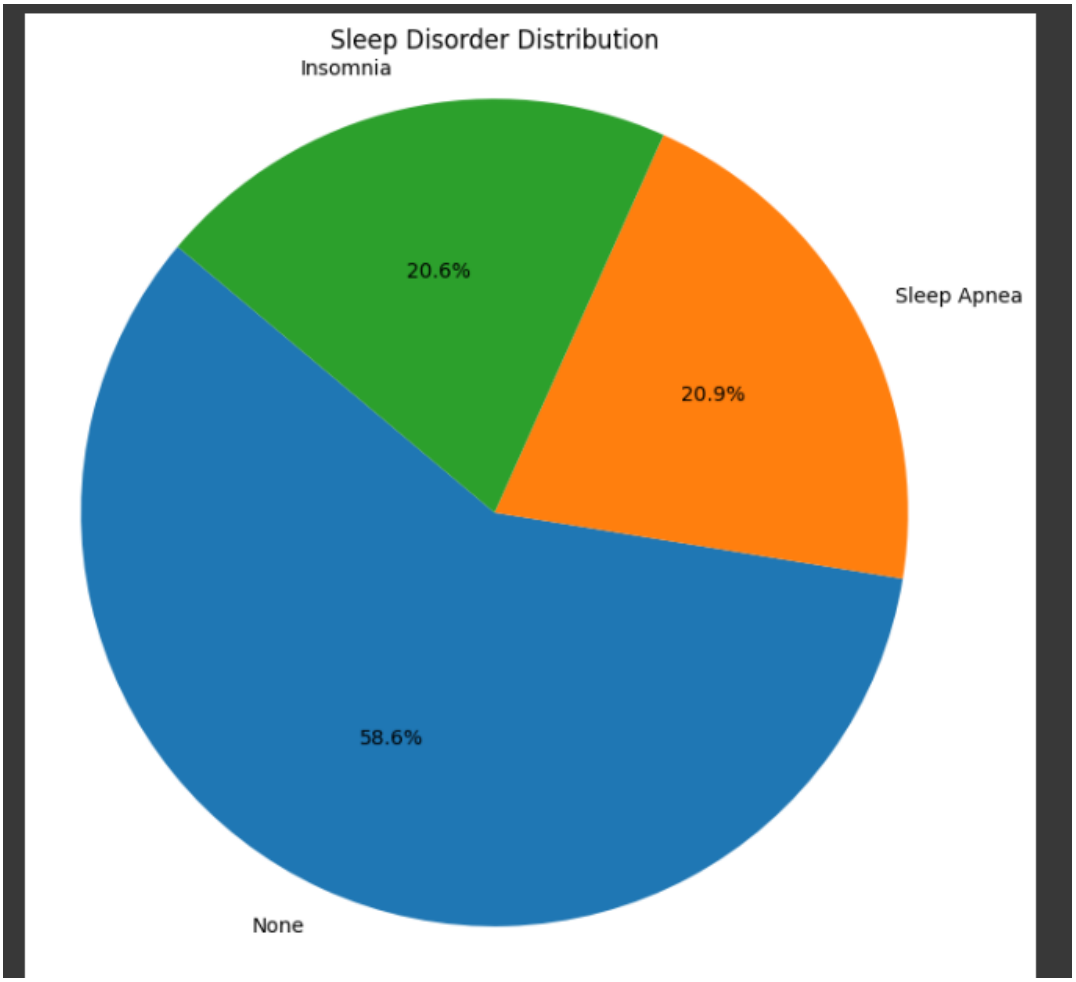


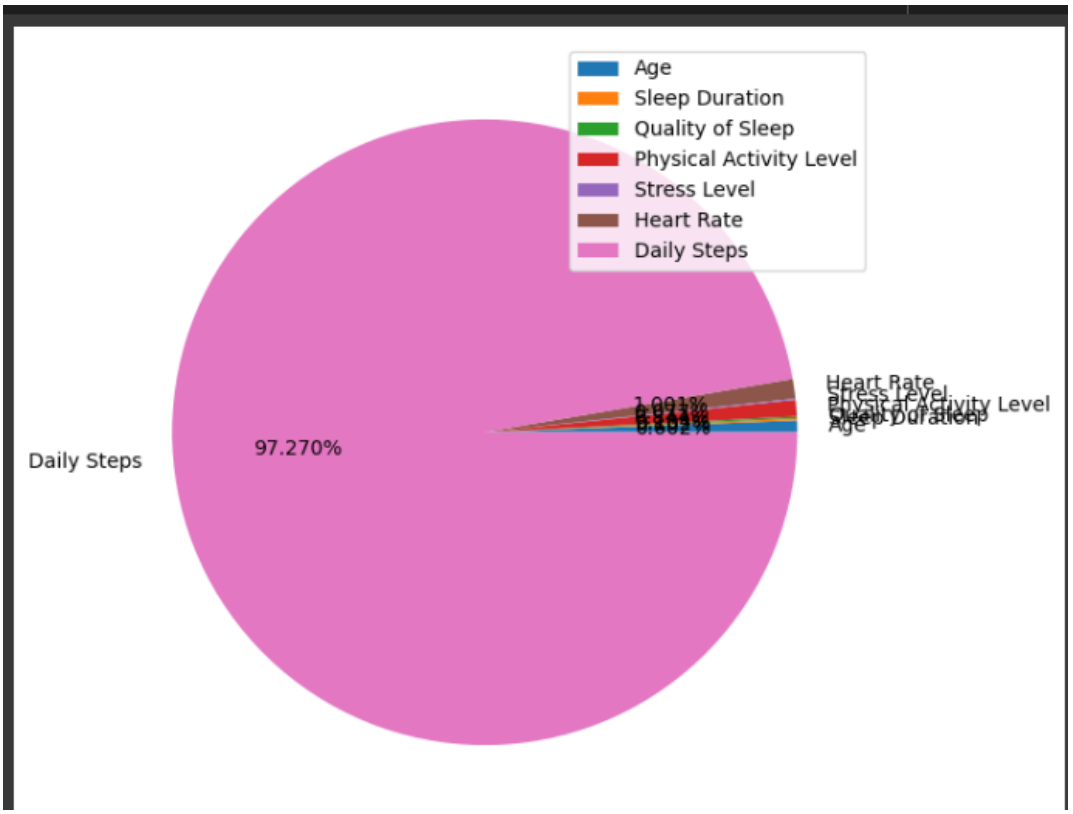


```
sns.countplot(x='Sleep Disorder', data=df)
```

```
<Axes: xlabel='Sleep Disorder', ylabel='count'>
```

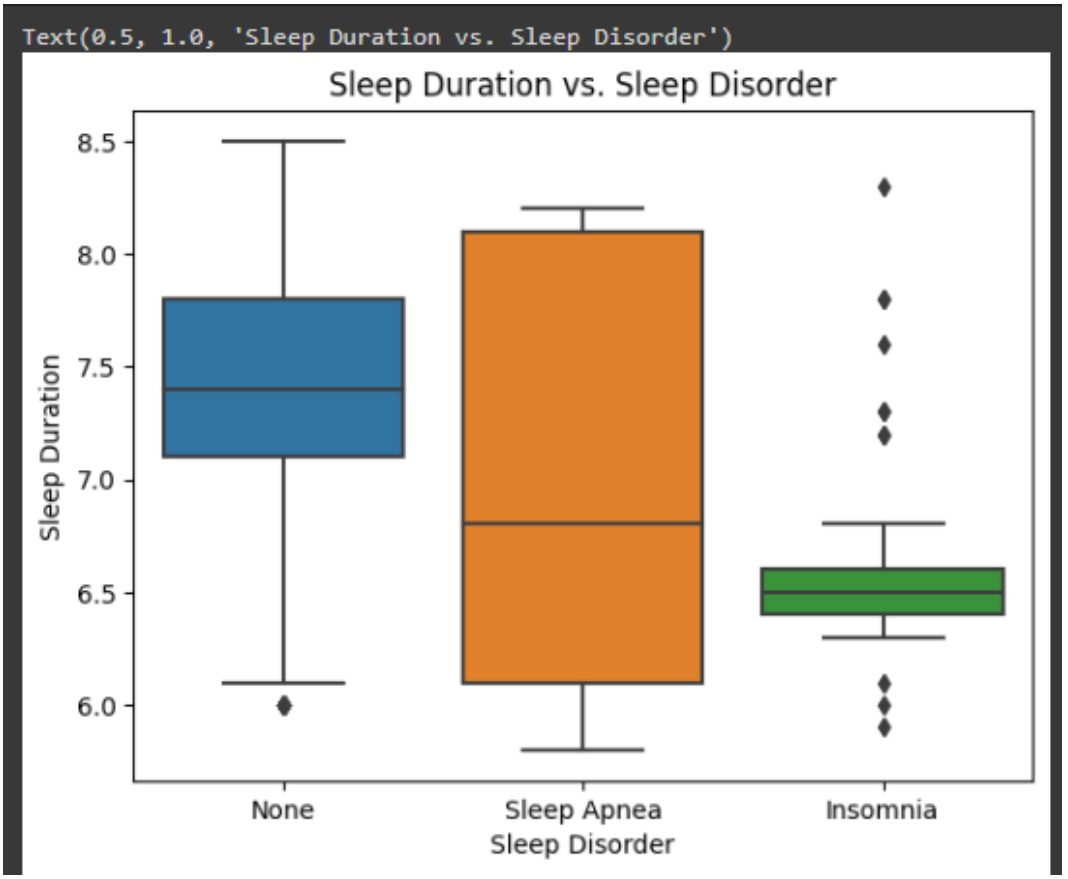






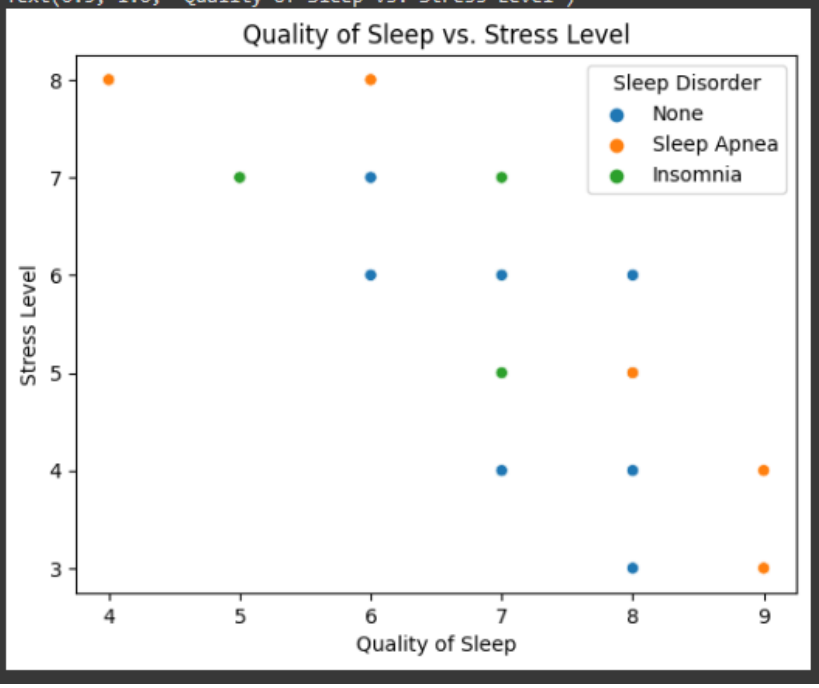
Activity 2.2: Bivariate analysis

To find the relation between two features we use bivariate analysis.

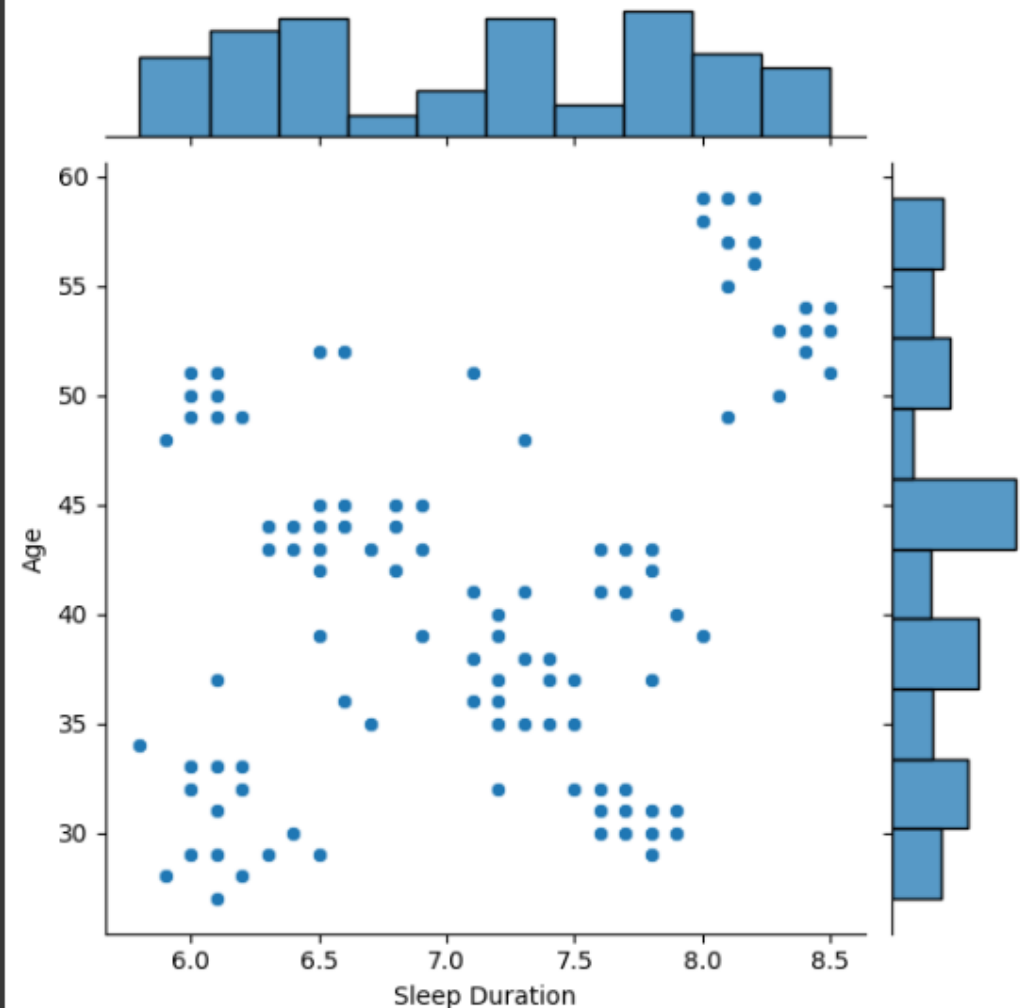


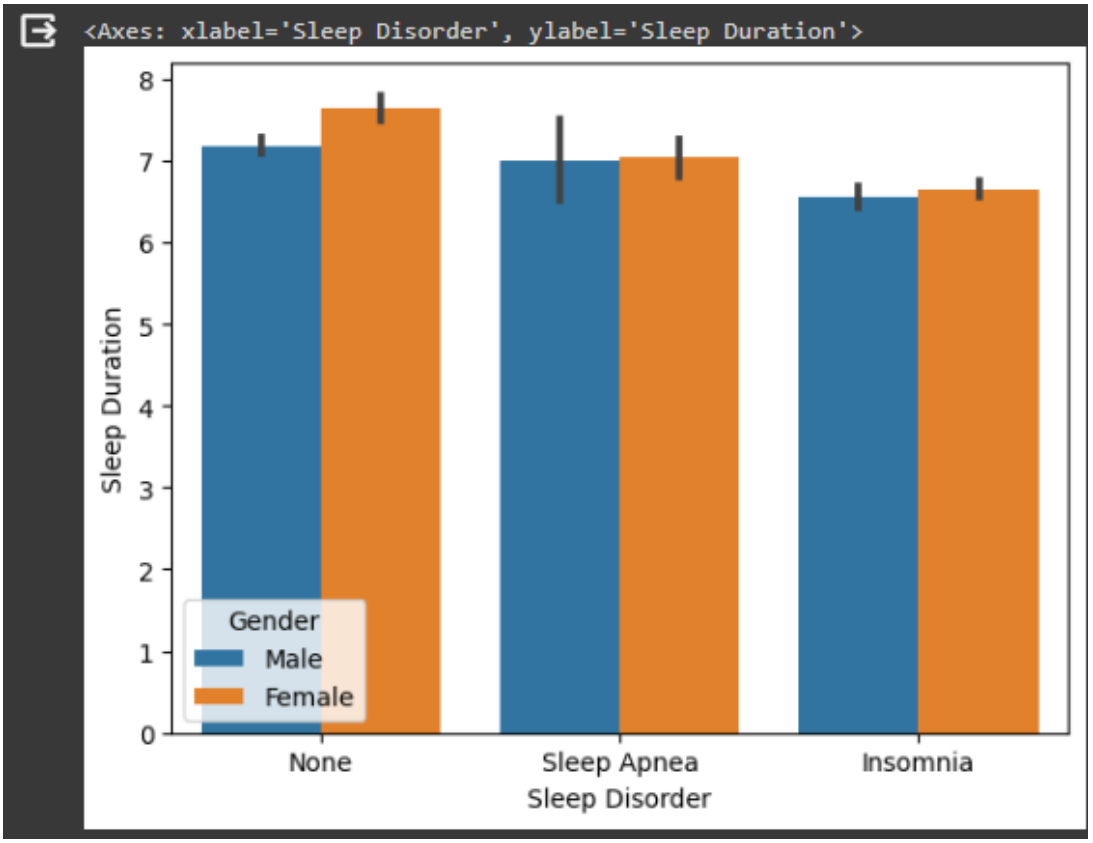
```
sns.scatterplot(x='Quality of Sleep', y='Stress Level', hue='Sleep Disorder', data=df)
plt.title('Quality of Sleep vs. Stress Level')
```

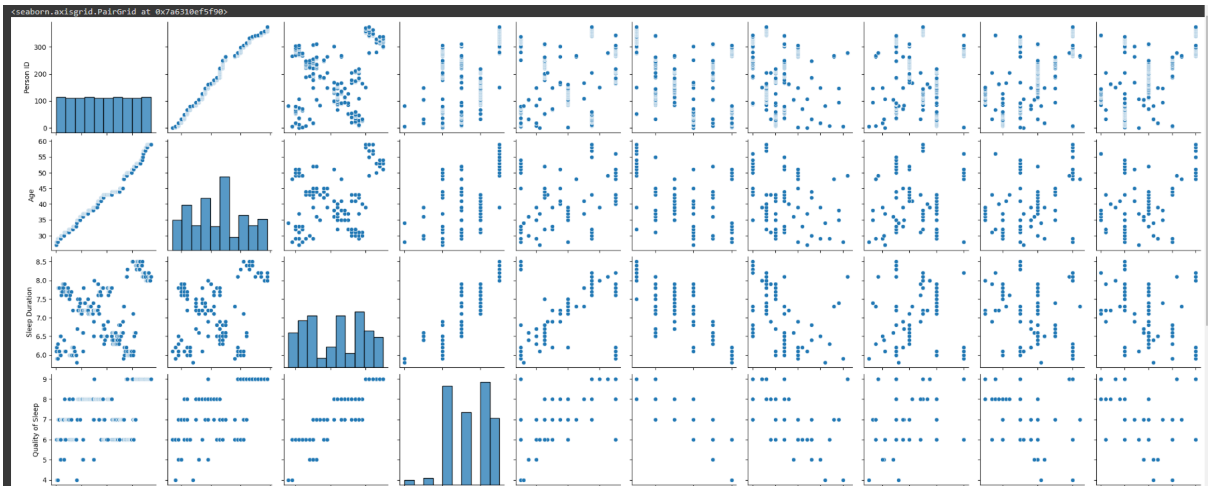
```
Text(0.5, 1.0, 'Quality of Sleep vs. Stress Level')
```

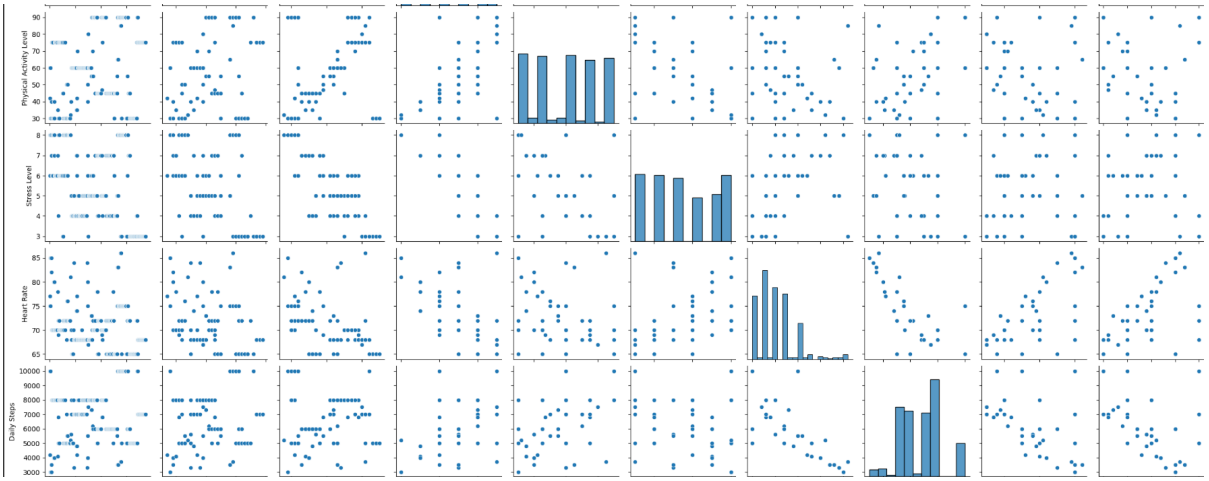


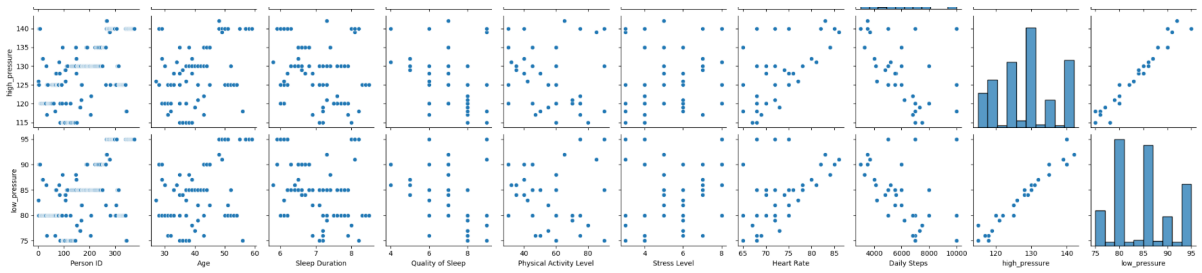

```
<seaborn.axisgrid.JointGrid at 0x7a63118cbe20>
```



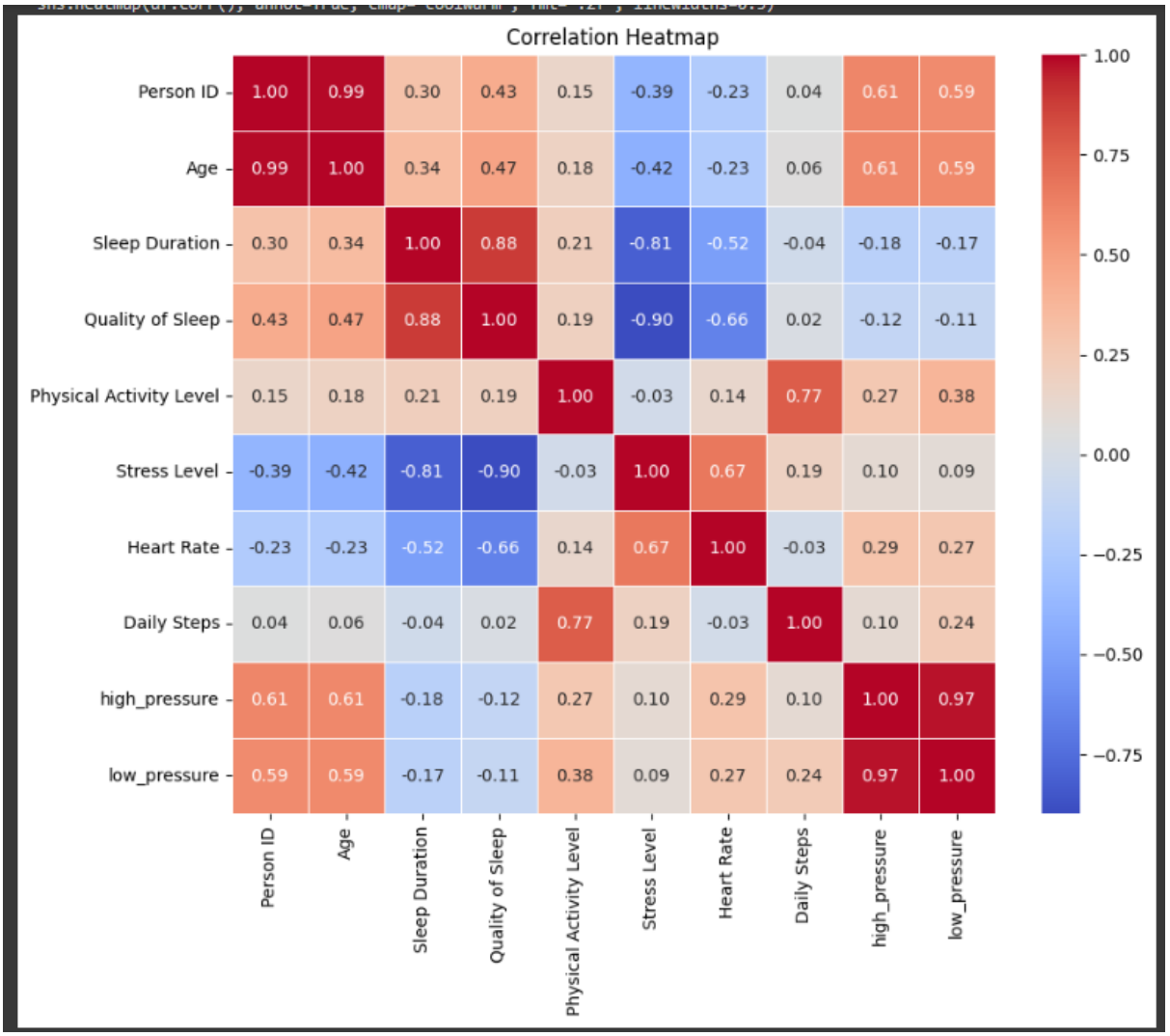

<Axes: xlabel='Sleep Disorder', ylabel='Sleep Duration'>
Sleep Duration
Gender
Male
Female
None
Sleep Apnea
Insomnia
Sleep Disorder


<seaborn.axisgrid.PairGrid at 0x7a6310ef5f90>
Person ID
Age
Sleep Duration
Quality of Sleep


Physical Activity Level
Stress Level
Heart Rate
Daily Steps



Activity 2.3: Multivariate analysis



Feature Selection:

```
df.drop(columns=["Quality of Sleep","low_pressure","Person ID"],axis=1,inplace=True)
```

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

	Gender	Age	Occupation	Sleep Duration	Physical Activity Level	Stress Level	BMI Category	Heart Rate	Daily Steps	Sleep Disorder	high_pressure
0	1	27	9	6.1	42	6	1	77	4200	1	126
1	1	28	1	6.2	60	8	0	75	10000	1	125
2	1	28	1	6.2	60	8	0	75	10000	1	125
3	1	28	6	5.9	30	8	1	85	3000	2	140
4	1	28	6	5.9	30	8	1	85	3000	2	140

Splitting data into train and test :

First split the dataset into x and y and then split the data set

Defining independent and dependent variables(x,y)

```
[58] x=df.drop(columns=["Sleep Disorder"])
     y=df.iloc[:,9:10]
```

```
[59] x.head()
```

	Gender	Age	Occupation	Sleep Duration	Physical Activity Level	Stress Level	BMI Category	Heart Rate	Daily Steps	high_pressure
0	1	27	9	6.1	42	6	1	77	4200	126
1	1	28	1	6.2	60	8	0	75	10000	125
2	1	28	1	6.2	60	8	0	75	10000	125
3	1	28	6	5.9	30	8	1	85	3000	140
4	1	28	6	5.9	30	8	1	85	3000	140

```
[60] y.head()
```

	Sleep Disorder
0	1
1	1
2	1
3	2
4	2

```
x["BMI Category"].value_counts()
```

```
0    216
1    158
Name: BMI Category, dtype: int64
```

```
[62] x["Gender"].value_counts()
```

```
1    189
0    185
Name: Gender, dtype: int64
```

```
[63] x["Occupation"].value_counts()
```

```
5     73
1     71
2     63
3     47
10    40
0     37
7     32
9      4
8      4
6      2
4      1
Name: Occupation, dtype: int64
```

```
[64] y['Sleep Disorder']
```

```
0      1
1      1
2      1
3      2
4      2
      ..
369    2
370    2
371    2
372    2
373    2
Name: Sleep Disorder, Length: 374, dtype: int64
```

Encoding:

```
[52] from sklearn.preprocessing import LabelEncoder
     le=LabelEncoder()
     df["Gender"]=le.fit_transform(df["Gender"])
     df['Occupation'] = le.fit_transform(df['Occupation'])
     df['BMI Category'] = le.fit_transform(df['BMI Category'])
     df['Sleep Disorder'] = le.fit_transform(df['Sleep Disorder'])
```

```
df.head()
```

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Heart Rate	Daily Steps	Sleep Disorder	high_pressure	low_pressure
0	1	1	27	9	6.1	6	42	6	1	77	4200	1	126	83
1	2	1	28	1	6.2	6	60	8	0	75	10000	1	125	80
2	3	1	28	1	6.2	6	60	8	0	75	10000	1	125	80
3	4	1	28	6	5.9	4	30	8	1	85	3000	2	140	90
4	5	1	28	6	5.9	4	30	8	1	85	3000	2	140	90

Train Test Split

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

```
[67] x_train.shape,x_test.shape,y_train.shape,y_test.shape

     ((299, 10), (75, 10), (299, 1), (75, 1))
```

Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

1. Logistic Regression :

```
[68] from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

[69] logistic_regression_model = LogisticRegression()
```

```
logistic_regression_model.fit(x_train, y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning:
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarnin
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

```
[71] y_pred = logistic_regression_model.predict(x_test)

[72] y_pred

array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 2, 1, 1, 1, 2, 0, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2,
       1, 1, 2, 2, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 2, 0])
```

```
y_test
```

	Sleep Disorder
122	1
295	2
311	0
210	1
204	1
...	...
173	1
56	1
8	1
361	2
298	1

75 rows × 1 columns

2. XGB boost :

```
[78] import xgboost as xgb

[79] xgb_model = xgb.XGBClassifier()

xgb_model.fit(x_train, y_train)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, objective='multi:softprob', ...)
```

```
[81] y_pred1 = xgb_model.predict(x_test)

[82] y_pred1
```

```
array([1, 2, 0, 1, 1, 1, 0, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 2, 1, 2, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 2, 0, 2,
       1, 1, 2, 2, 2, 1, 0, 1, 0, 1, 1, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 2, 0, 0, 1, 1, 1, 2, 1])
```

3. Decision Tree :

```
from sklearn.tree import DecisionTreeClassifier
tree_model = DecisionTreeClassifier(random_state=42)
tree_model.fit(x_train, y_train)
```

```
DecisionTreeClassifier(random_state=42)
```

```
[88] y_pred_tree = tree_model.predict(x_test)
     y_pred_tree
```

```
array([1, 2, 0, 1, 1, 1, 0, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 2, 1, 2, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 2, 0, 2,
       1, 1, 2, 2, 2, 1, 0, 1, 0, 1, 1, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 2, 0, 0, 1, 1, 1, 2, 1])
```

4. Random Forest :

```
[93] from sklearn.ensemble import RandomForestClassifier
     from sklearn.model_selection import GridSearchCV
```

```
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
[95] random_forest_model.fit(x_train, y_train)
```

```
<ipython-input-95-bff935996b53>:1: DataConversionWarning: A column-vector y was pass
  random_forest_model.fit(x_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```
[96] y_pred = random_forest_model.predict(x_test)
```

```
[97] y_pred
```

```
array([1, 2, 0, 1, 1, 1, 0, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 2, 1, 2, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 2, 0, 2,
       1, 1, 2, 2, 2, 1, 0, 1, 0, 1, 1, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 2, 0, 0, 1, 1, 1, 2, 1])
```

5. Random Forest (Hyper Tuning Parameter) :

```
[102] param_grid = {
          'n_estimators': [100, 200, 300],
          'max_depth': [4, 6, 8],
          'min_samples_split': [2, 5, 10],
          'min_samples_leaf': [1, 2, 4],
          'max_features': ['auto', 'sqrt', 'log2']
      }
```

```
[103] grid_search = GridSearchCV(estimator=random_forest_model, param_grid=param_grid, cv=5, scoring='accuracy', verbose=2)

[104] grid_search.fit(x_train, y_train)
      estimator.fit(X_train, y_train, **fit_params)
    [CV] END max_depth=8, max_features=log2, min_samples_leaf=4, min_samples_split=5, n_estimators=300; total time=   0.5s
    [CV] END max_depth=8, max_features=log2, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=   0.2s
    /usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was pa
      estimator.fit(X_train, y_train, **fit_params)
    /usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was pa
      estimator.fit(X_train, y_train, **fit_params)
    [CV] END max_depth=8, max_features=log2, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=   0.2s
    [CV] END max_depth=8, max_features=log2, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=   0.2s
    /usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was pa
      estimator.fit(X_train, y_train, **fit_params)
    /usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was pa
      estimator.fit(X_train, y_train, **fit_params)
    [CV] END max_depth=8, max_features=log2, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=   0.2s
    [CV] END max_depth=8, max_features=log2, min_samples_leaf=4, min_samples_split=10, n_estimators=100; total time=   0.2s
```

```
[CV] END max_depth=8, max_features=log2, min_samples_leaf=4, min_samples_split=10, n_estimators=300; total time=   0.5s
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:909: DataConversionWarning: A column-vector y was pa
  self.best_estimator_.fit(X, y, **fit_params)
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'` has been deprecat
  warn(
```

GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier

```
[105] best_params = grid_search.best_params_
      print("Best Hyperparameters:", best_params)

      Best Hyperparameters: {'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
```

```
[106] best_random_forest_model = grid_search.best_estimator_

[107] y_pred = best_random_forest_model.predict(x_test)
```

Milestone 5: Performance Testing & Hyperparameter Tuning

Activity1: Testing model with multiple evaluation metrics

1. Logistic Regression :

```
[74] accuracy = accuracy_score(y_test, y_pred)
     conf_matrix = confusion_matrix(y_test, y_pred)
     classification_rep = classification_report(y_test, y_pred)

[75] accuracy

     0.6666666666666666

[76] confusion_matrix(y_test,y_pred)

     array([[ 7, 13,  0],
            [ 3, 37,  1],
            [ 0,  8,  6]])

     print(classification_rep)

                   precision    recall  f1-score   support

                0       0.70      0.35      0.47        20
                1       0.64      0.90      0.75        41
                2       0.86      0.43      0.57        14

         accuracy                           0.67        75
        macro avg       0.73      0.56      0.60        75
     weighted avg       0.70      0.67      0.64        75
```

2. XGB boost :

```
     accuracy_score(y_test,y_pred1)

     0.9066666666666666

[85] confusion_matrix(y_test,y_pred1)

     array([[16,  2,  2],
            [ 1, 40,  0],
            [ 1,  1, 12]])

[86] print(classification_report(y_test,y_pred1))

                   precision    recall  f1-score   support

                0       0.89      0.80      0.84        20
                1       0.93      0.98      0.95        41
                2       0.86      0.86      0.86        14

         accuracy                           0.91        75
        macro avg       0.89      0.88      0.88        75
     weighted avg       0.91      0.91      0.91        75
```

3. Decision Tree :

```
[89] accuracy_tree = accuracy_score(y_test, y_pred_tree)
     print("Decision Tree Accuracy:", accuracy_tree)

     Decision Tree Accuracy: 0.9066666666666666

     classification_report_tree = classification_report(y_test, y_pred_tree)
     print("Decision Tree Classification Report:\n", classification_report_tree)

     Decision Tree Classification Report:
                   precision    recall  f1-score   support

                0       0.89      0.80      0.84        20
                1       0.93      0.98      0.95        41
                2       0.86      0.86      0.86        14

         accuracy                           0.91        75
        macro avg       0.89      0.88      0.88        75
     weighted avg       0.91      0.91      0.91        75

[91] confusion_matrix(y_test,y_pred_tree)

     array([[16,  2,  2],
            [ 1, 40,  0],
            [ 1,  1, 12]])
```

```
     probability=tree_model.predict_proba(x_test)[:,1]
     probability

     array([1.        , 0.1       , 0.        , 1.        , 1.        ,
            0.66666667, 0.        , 0.        , 0.75      , 1.        ,
            1.        , 1.        , 1.        , 0.        , 0.        ,
            0.7       , 1.        , 1.        , 1.        , 1.        ,
            1.        , 1.        , 0.        , 0.        , 1.        ,
            0.75      , 0.        , 0.        , 0.        , 1.        ,
            0.        , 1.        , 1.        , 0.33333333, 1.        ,
            0.        , 1.        , 1.        , 1.        , 1.        ,
            0.75      , 0.        , 0.        , 0.16666667, 1.        ,
            0.66666667, 0.        , 0.25      , 0.        , 1.        ,
            0.        , 1.        , 0.        , 1.        , 0.875     ,
            0.        , 1.        , 1.        , 1.        , 1.        ,
            0.        , 0.        , 0.33333333, 0.        , 0.        ,
            1.        , 1.        , 0.        , 0.33333333, 0.        ,
            1.        , 0.875     , 1.        , 0.        , 1.        ])
```

4. Random Forest :

```
[98] accuracy = accuracy_score(y_test, y_pred)
     conf_matrix = confusion_matrix(y_test, y_pred)
     classification_rep = classification_report(y_test, y_pred)

[99] print(f"Accuracy: {accuracy}")

     Accuracy: 0.9066666666666666
```

```
print("Confusion Matrix:")
print(conf_matrix)

Confusion Matrix:
[[16  2  2]
 [ 1 40  0]
 [ 1  1 12]]

[101] print("Classification Report:")
      print(classification_rep)

      Classification Report:
                    precision    recall  f1-score   support

                 0       0.89      0.80      0.84        20
                 1       0.93      0.98      0.95        41
                 2       0.86      0.86      0.86        14

          accuracy                           0.91        75
         macro avg       0.89      0.88      0.88        75
      weighted avg       0.91      0.91      0.91        75
```

5. **Random Forest (Hyper Tuning Parameter) :**

```
[109] print(f"Accuracy: {accuracy}")

      Accuracy: 0.9066666666666666

      print("Confusion Matrix:")
      print(conf_matrix)

      Confusion Matrix:
      [[16  2  2]
       [ 1 40  0]
       [ 1  1 12]]

[111] print("Classification Report:")
      print(classification_rep)

      Classification Report:
                    precision    recall  f1-score   support

                 0       0.89      0.80      0.84        20
                 1       0.93      0.98      0.95        41
                 2       0.86      0.86      0.86        14

          accuracy                           0.91        75
         macro avg       0.89      0.88      0.88        75
      weighted avg       0.91      0.91      0.91        75
```

Activity 1.1: Compare the model

Analyzing the accuracy using different evaluation metrics:

Logistic Regression	XGB boost	Decision Tree	Random Forest	Random Forest (Hyper Parameter Tuning)
0.66666	0.90666	0.90666	0.90666	0.90666

The table provides an overview of the classification accuracy scores for different machine learning models applied to the Sleep Oracle dataset. The metrics represent the performance of Logistic Regression, XGB Boost, Decision Tree, and Random Forest models, with an additional entry for Random Forest after hyperparameter tuning.

The results indicate that all models, including the initial Random Forest, demonstrate strong predictive capabilities with accuracy scores consistently around 90.67%. This suggests that the baseline models already perform well in classifying sleep disorders based on the dataset features. Interestingly, the hyperparameter tuning of the Random Forest did not significantly improve its accuracy, maintaining the same level as the untuned version.

Milestone 6: Model Deployment

Activity 1: Save the best model

```
[113] import pickle

[114] pickle.dump(best_random_forest_model, open('model.pkl', 'wb'))
      model = pickle.load(open('model.pkl', 'rb'))
```

```
[115] import zipfile
      from zipfile import ZipFile

      file_name = '/content/templates.zip'

      with ZipFile(file_name, 'r') as zip:
        zip.extractall()
        print('Extracted Successfully')

      Extracted Successfully
```

```
[112] !pip install flask flask-ngrok

      Requirement already satisfied: flask in /usr/local/lib/python3.10/dist-packages (2.2.5)
      Requirement already satisfied: flask-ngrok in /usr/local/lib/python3.10/dist-packages (0.0.25)
      Requirement already satisfied: Werkzeug>=2.2.2 in /usr/local/lib/python3.10/dist-packages (from flask) (3.0.1)
      Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.10/dist-packages (from flask) (3.1.2)
      Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.10/dist-packages (from flask) (2.1.2)
      Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.10/dist-packages (from flask) (8.1.7)
      Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from flask-ngrok) (2.31.0)
      Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2>=3.0->flask) (2.1.3)
      Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->flask-ngrok) (3.3.2)
      Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->flask-ngrok) (3.4)
      Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->flask-ngrok) (2.0.7)
      Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->flask-ngrok) (2023.7.22)
```

```
[116] !pip install pyngrok

      Requirement already satisfied: pyngrok in /usr/local/lib/python3.10/dist-packages (7.0.1)
      Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages (from pyngrok) (6.0.1)
```

Activity 2: Integrate with Web Framework

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

Activity 2.1: Building Html Pages:

For this project the HTML file used is:

- index.html

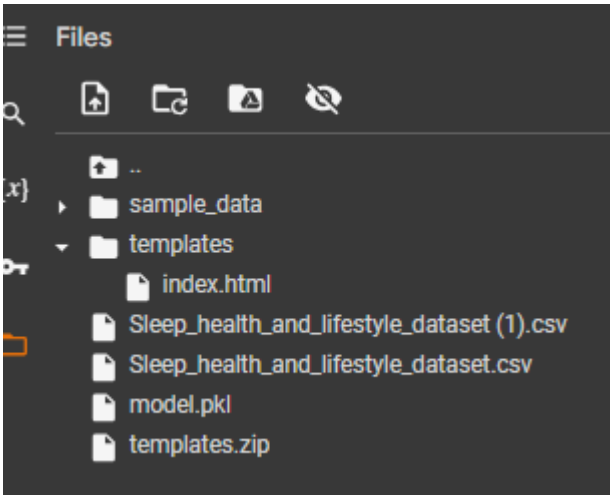
index.html

```
<!DOCTYPE html>
<html>
<head>
<title> Sleep Disorder Prediction</title>
<style>
    body {
    text-align: center;
    background-image: url('https://t4.ftcdn.net/jpg/02/43/11/81/360_F_243118100_iDxc7BO4YmaSKwhMTcleQVuoYVhAnSUM.jpg');
    background-repeat: no-repeat;
    background-size: cover;
    background-attachment: fixed;
    /* color: white; */
    }

    h1{
        color: rgb(215, 218, 250);
    }

    input{
        margin-top: 10px;
        width: 300px;
        height: 30px;
        margin-bottom: 10px;
    }
    select{
        width: 300px;
        height: 35px;
        margin-top: 10px;
        margin-bottom: 10px;
    }
    table{
        align-items: center;
        border-spacing:20px;
    }
```

```
<center><h1><b><i><font size=15>Sleep Disorder Prediction</font></i></b></h1></center>
<div style="background-color:white">
<hr></div>
<h4>
    <form action="{{url_for('predict')}}" method="post">
     <center>
     <table  cellpadding="7" width="90%" style="margin-top: -20px;">
        <tr>
            <td >
                Gender :
                <br>
                <input type='number' name='gender' min="0" max="1" placeholder='Enter 1 for male and 0 for female' required='required' />
                <!-- <select name="dropdown">
                    <option value="select">Enter 1 for male and 0 for female</option>
                    <option value="Male">1</option>
                    <option value="Female">0</option>
                </select> -->
            </td>

            <td >
                Age :
                <br />
                    <input type='number' name='Age' placeholder='Enter age in years only' required='required' />
            </td>

            <td >
                Occupation :
                <br>
                <input type='number' name='Occupation' min="0" max="9" required='required' />
                        <!-- <select name="dropdown">
                            <option value="select">Select Occupation</option>
                            <option value="Accountant">0</option>
                            <option value="Doctor">1</option>
                            <option value="Engineer">2</option>
                            <option value="Lawyer">3</option>
                            <option value="Manager">4</option>
                            <option value="Nurse">5</option>
                            <option value="Sales Representative">6</option>
                            <option value="Salesperson">7</option>
                            <option value="Scientist">8</option>
                            <option value="Software Engineer">9</option>
                            <option value="Teacher">10</option>
                        </select> -->
            </td>
```

```
            <td>
                BMI Category :
                <input type='number' name='BMI' min="0" max="1" placeholder='Enter 1 for Overweight and 0 for Normal' required='required' />
            </td>

            <td>
                Blood Pressure :
                <input type='number' name='Blood Pressure' placeholder='systolic' required='required' />
            </td>

            <td>
                Heart Rate :
                <br>
                <input type='number' name='Heart Rate' placeholder='beats per minute(bpm)' required='required' />
            </td>
        </tr>

        <tr>
            <td >
                Daily Steps :
                <input type='number' name='Daily Steps' required='required' />
            </td>
        </tr>
    </table>
</center>
    <button type="submit"><b>Predict</b></button>
</form>
</h4>
<h2>
<b>{{ prediction_text }}</b>
</h2>
```

Activity 2.2: Build Python code:



Import the libraries

```
import flask
from flask import Flask, render_template, request
import pickle
import numpy as np
from flask_ngrok import run_with_ngrok
import warnings
warnings.filterwarnings('ignore')
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

Load the model:

```
app = Flask(__name__)
run_with_ngrok(app)

model = pickle.load(open('model.pkl', 'rb'))
```

Render HTML page:

```python
@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['GET', "POST"])
def predict():
    if request.method=='POST':
        # input_values = [float(x) for x in request.form.values()]
        input_values = [x for x in request.form.values()]
        inp_features = [input_values]
        prediction = model.predict(inp_features)
        if prediction == 0:
            return render_template('index.html', prediction_text='Person has Insomnia')
        elif prediction == 1:
            return render_template('index.html', prediction_text='No sleep disorder')
        elif prediction==2:
            return render_template('index.html', prediction_text='Person have Sleep Apnea')
app.run()
```

Activity 2.3: Run the web application

```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
 * Running on http://d448-34-23-147-78.ngrok-free.app
 * Traffic stats available on http://127.0.0.1:4040
```

Predictions:

```
Select Occupation


"Accountant"            " 0 "
"Doctor"                " 1 "
"Engineer"              " 2 "
"Lawyer"                " 3 "
"Manager"               " 4 "
"Nurse"                 " 5 "
"Sales Representative"  " 6 "
"Salesperson"           " 7 "
"Scientist"             " 8 "
"Software Engineer"     " 9 "
"Teacher"               " 10 "
```


Sleep Disorder Prediction
Gender :
Enter 1 for male and 0 for female
Age :
Enter age in years only
Occupation :
Sleep Duration :
Enter Sleep duration in hours
Physical activity level :
Enter in minutes/day
Stress level :
BMI Category :
Enter 1 for Overweight and 0 for Normal
Blood Pressure :
systolic
Heart Rate :
beats per minute(bpm)
Daily Steps :
Predict


Sleep Disorder Prediction
Gender :
1
Age :
31
Occupation :
1
Sleep Duration :
7.7
Physical activity level :
75
Stress level :
6
BMI Category :
0
Blood Pressure :
120
Heart Rate :
70
Daily Steps :
8000
Predict


Sleep Disorder Prediction
Gender :
Enter 1 for male and 0 for female
Age :
Enter age in years only
Occupation :
Sleep Duration :
Enter Sleep duration in hours
Physical activity level :
Enter in minutes/day
Stress level :
BMI Category :
Enter 1 for Overweight and 0 for Normal
Blood Pressure :
systolic
Heart Rate :
beats per minute(bpm)
Daily Steps :
Predict
Person have Sleep Apnea

Milestone 7: Project Demonstration & Documentation

Activity 1:- Record explanation Video for project end to end solution

Drive Link:

https://drive.google.com/file/d/15ThCYQeUl_33AwwiXotbJFX2TbxPG2C_/view?usp=sharing

Activity 2:- Project Documentation-Step by step project development procedure:

Link: Project Report Documentation