A red logo with a building

AI-generated content may be incorrect.

Data Science Programming

Report

Team:

**Farah Ahmad Mustafa**

**Nagham Thamer Alotaibi**

**\**

Supervised By:

**Dr. Mohammed Eshtay**

Semester:

2025/2026

Second Semester

***Abstract:***

This project aims to develop a machine learning model to detect potential sleep disorders such as insomnia and sleep apnea based on a person's lifestyle, demographic, and physiological indicators. Using a real-world dataset of sleep health and lifestyle attributes, we will preprocess the data using Pandas, engineer relevant features, and apply Scikit-learn classification algorithms to predict sleep disorder types. The project also includes JSON file manipulation to manage model outputs and user inputs, and web scraping to enrich the dataset with sleep-related health information. Multithreading will be implemented to optimize processing performance.

***Introduction:***

This report details the analysis of a dataset focused on sleep health and lifestyle factors. The primary objective of this project was to explore the relationships between various demographic, health, and lifestyle variables and their potential impact on sleep patterns and overall well-being. The analysis involved rigorous data preprocessing and engineering steps, followed by comprehensive exploratory data analysis (EDA) to uncover insights and patterns within the data. The dataset, sourced from Kaggle, provides a rich foundation for understanding how factors like age, occupation, physical activity, stress levels, and physiological metrics correlate with sleep duration, sleep quality, and the presence of sleep disorders.

## ***Dataset Description:***

This analysis is based on a dataset obtained from Kaggle, originally consisting of 405 individual records across 13 distinct columns. The dataset captures a wide range of information related to sleep health and associated lifestyle factors. Key variables include demographic details such as Person ID (a unique identifier), Gender, Age, and Occupation. The core of the dataset revolves around health and lifestyle metrics, including Sleep Duration (in hours), Quality of Sleep (rated on a scale from 1 to 10), Physical Activity Level (measured in minutes per day), Stress Level (also on a 1 to 10 scale), and BMI Category (classifying individuals as Underweight, Normal, Overweight, or Obese). Additional variables include Blood Pressure, which is recorded as a single text string in systolic/diastolic format (e.g., "124/70"), Heart Rate (in beats per minute), Daily Steps, and the presence or absence of a diagnosed sleep disorder, such as Insomnia or Sleep Apnea.

The dataset contains a mix of data types, including numerical values (e.g., age, heart rate), categorical labels (e.g., gender, occupation), and textual entries (e.g., blood pressure). Upon initial inspection, several data quality issues were identified that required preprocessing. Notably, missing values were found in columns such as "Sleep Duration," "Quality of Sleep," and "Daily Steps." Additionally, the Blood Pressure column, stored as a text string, needed to be parsed into separate numerical values for effective analysis.

Overall, the dataset provides a valuable opportunity to explore the relationships between sleep health and various lifestyle factors, including stress levels, physical activity, and BMI.

***Columns information:***

**Person ID:** A unique identifier for everyone.

**Gender:** The gender of the individual (e.g., Male, Female).

**Age:** The age of the individual in years**.**

**Occupation:** The profession or employment status of the individual.

**Sleep Duration (hours):** The reported average sleep duration per night, measured in hours. (Contains some missing values.)

**Quality of Sleep (scale: 1-10):** A subjective rating of sleep quality on a scale from 1 (poor) to 10 (excellent). (Contains some missing values.)

**Physical Activity Level (minutes/day):** The average daily duration of physical activity, measured in minutes.

**Stress Level (scale: 1-10):** A subjective rating of stress level on a scale from 1 (low) to 10 (high).

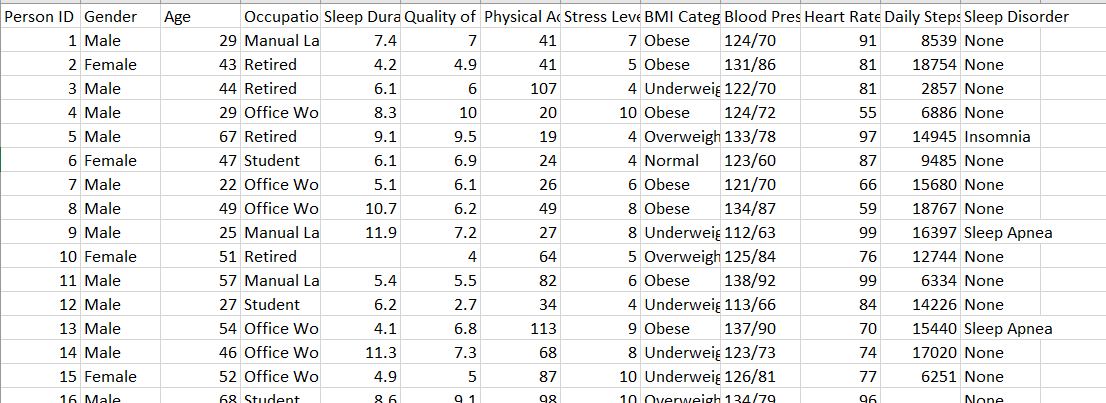
**BMI Category:** A classification of individuals based on their Body Mass Index (BMI), such as Underweight, Normal, Overweight, or Obese**.**

**Blood Pressure (systolic/diastolic**): The blood pressure readings of the individual in the format "systolic/diastolic".

**Heart Rate (bpm):** The resting heart rate of the individual, measured in beats per minute.

**Daily Steps:** The average number of steps taken per day. (Contains some missing values.)

**Sleep Disorder:** Indicates whether the individual has been diagnosed with a sleep disorder (e.g., Insomnia, Sleep Apnea, or None).

**here is a screenshot of the data in excel:**

## ***Data Preprocessing and Engineering:***

A systematic data preprocessing and engineering pipeline was implemented to address the identified data quality issues and prepare the dataset for robust analysis. The process began with loading the primary dataset (sleep\_health\_lifestyle\_with\_issues.csv) and an auxiliary dataset (new.csv, which used ‘No’ instead of ‘None’ for sleep disorders and had duplicates) into Pandas DataFrames.

A significant early step involved handling the ‘Blood Pressure (systolic/diastolic)’ column. This string column was split into two distinct numerical columns, ‘Systolic’ and ‘Diastolic’, using the ‘/’ character as a delimiter. The pd.to\_numeric function was employed to convert these new columns to a numerical format, with the errors='coerce' argument ensuring that any non-numeric entries were replaced with Not-a-Number (NaN) values. Building upon this, two derived metrics were engineered. ‘Pulse Pressure’, calculated as the difference between systolic and diastolic pressure, was added as a new column. Additionally, a ‘BP Category’ column was created by applying a custom function (categorize\_bp) that classified individuals into ‘Hypotension’, ‘Normal’, ‘Elevated’, ‘Hypertension’, or ‘Unknown’ based on their systolic and diastolic readings. This structured approach transformed the raw blood pressure string into actionable numerical and categorical features. The intermediate DataFrame with these changes was saved temporarily.

Data integrity checks followed, starting with duplicate detection. An analysis of the second DataFrame (df2) revealed the presence of three duplicate rows, which were subsequently removed to prevent bias in the analysis, reducing the effective dataset size slightly.

Missing data imputation was addressed next. For the ‘Sleep Duration (hours)’ and ‘Quality of Sleep (scale: 1-10)’ columns, missing values were imputed using the mean value of the respective column. This strategy was chosen for its simplicity and ability to preserve the overall dataset size, although it can potentially reduce data variability.

Outlier detection and removal were performed using the Interquartile Range (IQR) method, a robust technique suitable for potentially non-normally distributed data. This method calculates the range between the 25th percentile (Q1) and the 75th percentile (Q3) and identifies values falling significantly outside this range (typically 1.5 times the IQR below Q1 or above Q3) as outliers. This process was applied to the ‘Daily Steps’, ‘Sleep Duration (hours)’, ‘Quality of Sleep (scale: 1-10)’, and ‘Heart Rate (bpm)’ columns to eliminate extreme or potentially erroneous values that could skew analytical results. Removing these outliers helps ensure that subsequent analyses reflect typical patterns within the data. The cleaned dataset post-imputation and outlier removal was saved.

Finally, categorical features needed to be converted into a numerical format suitable for many analytical techniques and machine learning algorithms. The LabelEncoder from the Scikit-learn library was utilized to transform the ‘Gender’, ‘Occupation’, ‘BMI Category’, ‘Sleep Disorder’, and ‘BP Category’ columns into numerical representations. Each unique category within these columns was assigned a distinct integer value. This encoding step standardized the data format across all columns, making the dataset ready for the exploratory analysis phase. The fully preprocessed and encoded dataset was saved for subsequent use.

***code:***

We applied first data engineering to preprocess the data so that it will not contain problems such as missing values , outliers , duplicates.

First we read the data using pandas library as csv

Then we realized that blood pressure is a problem because its written like this ###/## , it hard to analyze it and to let any code read it .

To fix this problem this should be converted to two columns .

**Code explanation :**

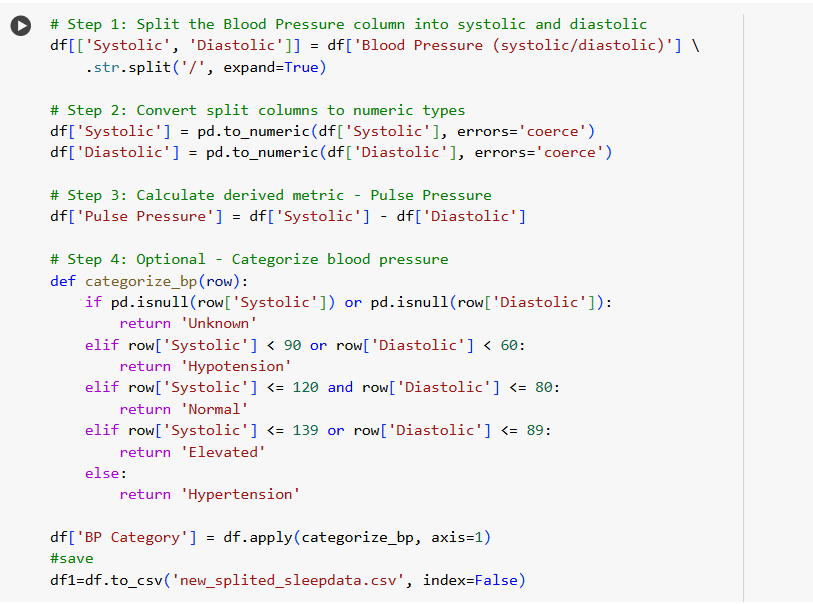
1-It extracts systolic and diastolic blood pressure components, calculates a derived metric called "Pulse Pressure," and categorizes individuals into blood pressure categories. And splits the "Blood Pressure (systolic/diastolic)" column into two new columns:

**Systolic**: The upper blood pressure value (measured during heartbeats).

**Diastolic:** The lower blood pressure value (measured between heartbeats).

2-The **str.split('/'):** function separates the values based on the / character.

3-expand=True ensures the split values are assigned to new columns



4-pd.to\_numeric: converts the column values from strings to numeric types.

5-errors='coerce' :replaces any invalid or non-numeric entries with NaN (missing values).

6-Then creates a new column, "Pulse Pressure," which is the difference between systolic and diastolic values.

**Pulse Pressure Formula**: Pulse Pressure = Systolic − Diastolic

7- Classifies each individual into one of four blood pressure categories based on systolic and diastolic values:

* **Hypotension: Low blood pressure.**
* **Normal: Ideal blood pressure.**
* **Elevated: Slightly higher than normal blood pressure.**
* **Hypertension: High blood pressure.**
* **If either value is missing, the category is set to "Unknown."**

**Method:**

A function (categorize\_bp) is applied row-wise using df.apply() to assess and assign categories.

8- Saves the processed dataset, including the derived metrics ("Pulse Pressure") and blood pressure categories ("BP Category"), as a new json file named new\_splited\_sleepdata.json to store the df1\_after\_split  
  
**Count missing values:**

A screenshot of a computer

AI-generated content may be incorrect.

Then we apply d2.describe: d2 is the same data set but we changed none to no in sleep disorder .

Descriptive Statistics of the Dataset

The describe() function provides a statistical summary of the numerical columns in the dataset. Below is an interpretation of the key statistics:

**Count:**

The number of non-missing values for each column.

**Mean:**

The average value for each column. For example:

Age: The average age of participants is approximately 39.78 years.

**Standard Deviation (std):**

A measure of the spread of data values from the mean. Higher values indicate greater variability. For example:

Heart Rate: The standard deviation is 15.15 bpm, suggesting moderate variation among participants.

**Minimum (min):**

The smallest value in each column. For example: Age: The youngest participant is 18 years old.

**25th Percentile (25%):**

The value below which 25% of the data falls. For instance:

Age: 25% of participants are younger than 29 years

**Median (50%):**

The middle value of the dataset, where 50% of values fall below and above it. For example:

Age: The median age is 39 years.

**75th Percentile (75%):**

The value below which 75% of the data falls. For example:

Physical Activity: 75% of participants engage in less than 94.5 minutes of physical activity daily.

**Maximum (max):**

The largest value in each column. For instance:

Age: The oldest participant is 90 years.

**Key Insights from Descriptive Statistics**

**Sleep and Activity:** On average, participants sleep for 8 hours and take approximately 10,867 steps daily, suggesting moderate physical activity .**Blood Pressure**: The systolic values range from 109 to 145 mmHg, and diastolic values range from 60 to 96 mmHg. Stress Levels: Stress levels vary widely, with a minimum of 1 and a maximum of 10, reflecting a diverse range of experiences.

**Age and Lifestyle:** The data includes participants across a broad age range (18 to 90), providing insights into sleep and lifestyle habits across different life stages.

Then we perform median , mode for the whole data set .

**Detecting duplicates:**

A computer screen shot of a program

AI-generated content may be incorrect.

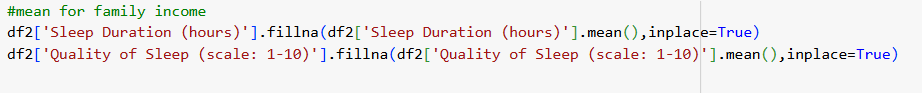
This code shows that there is 5 duplicated rows

A close-up of a computer code

AI-generated content may be incorrect.

Now it has 400 row

**Filling missing values :**





We choose mean for sleep duration and quality of sleep .This method is simple and preserves dataset size but may reduce variability in the data. And most suitable for this kind of information.

**A screenshot of a computer code

AI-generated content may be incorrect.Removing outliers:**

This code addresses outliers in specific columns of the dataset using the Interquartile Range (IQR) method. Outliers are extreme values that deviate significantly from the rest of the data, potentially distorting analyses.

Q1 (25th Percentile): The value below which 25% of the data falls.

Q3 (75th Percentile): The value below which 75% of the data falls.

IQR: The range between Q1 and Q3, representing the middle 50% of the data

**Purpose of Outlier Removal:**

Outliers can bias statistical analyses and machine learning models. Removing them ensures the dataset reflects typical values and patterns.

The IQR method is robust and suitable for detecting outliers in non-normal data distributions.

**Columns Processed:**

Daily Steps: Identifies individuals with extremely low or high physical activity levels.

Sleep Duration: Removes unrealistic sleep durations (e.g., excessively short or long).

Quality of Sleep: Excludes extreme self-reported values on the 1–10 scale.

Heart Rate: Eliminates abnormally low or high resting heart rates.

Impact on the Dataset:

Reduced Noise: By removing extreme values, the dataset becomes cleaner and more representative.

Improved Analysis: Ensures accurate calculations of central tendencies (mean, median) and relationships between variables.

Then save it to : df2**.**to\_json('new\_ aftermissing\_outliar\_sleepdata2.json')

**Encoding :**

**A screen shot of a computer code

AI-generated content may be incorrect.**

This code encodes categorical data into numeric values to make it suitable for analysis and machine learning algorithms. It uses the LabelEncoder from the sklearn.preprocessing module to handle all categorical columns in the dataset

**Purpose of Encoding:**

Machine learning algorithms typically require numerical inputs. Encoding categorical data ensures compatibility with these models.

Converts qualitative information (e.g., "Male" and "Female" in the Gender column) into numerical values for analysis.

**Columns Processed:**

Gender: Encoded as integers (e.g., 0 for "Male," 1 for "Female").

Occupation: Converts job categories into numeric codes.

BMI Category: Encodes health categories (e.g., "Underweight," "Normal," "Overweight").

Sleep Disorder: Assigns numeric values to sleep disorder statuses (e.g., "Yes" or "No").

BP Category: Transforms blood pressure classifications (e.g., "Normal," "Hypertension") into integers.

**Impact on Dataset:**

Standardization: All data is converted into a numerical format, making it ready for modeling.

Efficiency: Numeric representations improve computational efficiency for data processing and model training

df3= df2**.**to\_json('sleepdata\_encoded.json')saving it

df3\_to\_normalize = df3.copy()

***Exploratory Data Analysis (EDA)***

Following the preprocessing phase, an extensive Exploratory Data Analysis (EDA) was conducted to uncover patterns, relationships, and insights within the cleaned dataset. The EDA process involved statistical summaries, visualizations, and data aggregations.

Descriptive statistics provided an initial overview of the numerical variables. The .describe() method generated key metrics like count, mean, standard deviation, minimum, maximum, and quartile values for columns such as Age, Sleep Duration, Physical Activity Level, Stress Level, Heart Rate, Daily Steps, Systolic, Diastolic, and Pulse Pressure. This revealed, for instance, an average participant age of approximately 39.78 years and an average sleep duration of around 8 hours. The standard deviations indicated the degree of variability within each metric. Median and mode calculations further supplemented this understanding of central tendencies.

Visualizing the distribution of numerical variables was crucial. Histograms and Kernel Density Estimate (KDE) plots were generated for each numeric column. These plots helped assess the shape of the data distributions, identifying whether they approximated a normal distribution, exhibited skewness (indicating a concentration of values towards one end), or were multimodal (suggesting the presence of distinct subgroups within the data). Such visualizations are vital for selecting appropriate analytical methods and interpreting results correctly.

Correlation analysis was performed to understand the linear relationships between numerical variables. A correlation heatmap, generated using Seaborn, visually represented the Pearson correlation coefficients between all pairs of numeric columns. The heatmap used color intensity and annotations to show the strength and direction of correlations (ranging from -1 for perfect negative correlation to +1 for perfect positive correlation). This helped identify variables that tend to move together (e.g., Daily Steps and Physical Activity Level often show a positive correlation) or in opposite directions (e.g., higher Stress Level might correlate negatively with Quality of Sleep).

To delve deeper into specific relationships, scatter plots were employed. For example, a scatter plot of Systolic blood pressure versus Age was created to visually inspect potential trends or patterns between these two variables. Furthermore, a pairplot was generated, creating a matrix of scatter plots for every pairwise combination of numerical variables. This provided a comprehensive overview of bivariate relationships, with histograms along the diagonal showing the distribution of each individual variable.

Data aggregation and grouping techniques were used to explore differences across various subgroups. The data was grouped by ‘Gender’ to calculate the average ‘Sleep Duration (hours)’ for males and females separately. Similarly, the dataset was filtered to include only individuals older than 50, and then the average ‘Stress Level’ was calculated for this subgroup, further grouped by their ‘BP Category’. Another aggregation focused on ‘Occupation’, calculating the mean ‘Sleep Duration’, ‘Stress Level’, and ‘Physical Activity Level’ for each job category. These aggregations highlighted potential differences in lifestyle and health metrics based on demographic factors.

Visualizations were created to effectively communicate the findings from the aggregated data. Bar plots were used to compare the average sleep duration, stress level, and physical activity across different occupations, making it easy to identify potential occupational health trends. To analyze age-related patterns in sleep, participants were categorized into age groups (0-20, 20-40, 40-60, 60-80) using the pd.cut function. Box plots were then generated to show the distribution of ‘Sleep Duration’ within each age group, illustrating the median, quartiles, and potential outliers for each segment.

***A screenshot of a computer

AI-generated content may be incorrect.***

**Code Explanation :**

This code generates distribution plots for all numeric columns in the dataset (excluding the column "Blood Pressure (systolic/diastolic)"). The goal is to visualize how data in each column is distributed, which is critical for understanding the dataset and identifying patterns or anomalies.

**Purpose of Visualizing Distributions:**

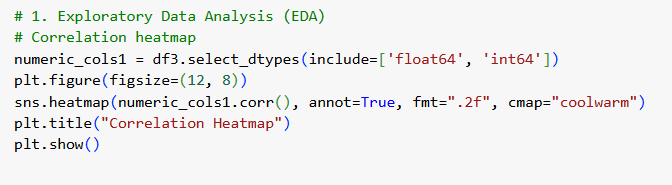
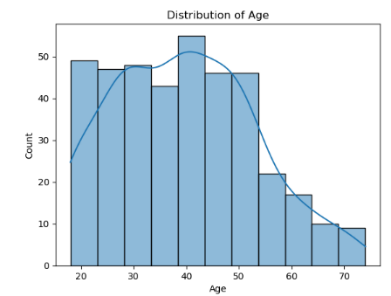
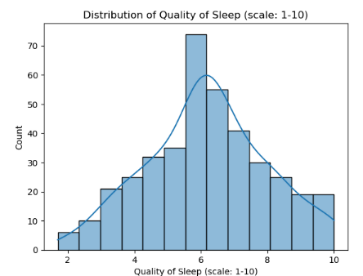
Understand Data Shape: Visualizations reveal whether the data follows a normal distribution, is skewed, or contains multiple peaks.

Identify Outliers: Helps detect extreme values that deviate significantly from the typical range.

Assess Data Quality: Flags issues such as gaps, uniform distributions, or irregular patterns.

**Insights for Dataset:**

Columns with symmetric, bell-shaped curves likely follow a normal distribution. Skewed distributions suggest the presence of extreme values or non-normality.Multimodal distributions indicate subgroups within the data.



This code generates a correlation heatmap for the numerical columns in the dataset. A correlation heatmap visually represents the relationships between numerical variables, aiding in the understanding of potential dependencies and interactions within the dataset. df3.select\_dtypes(include=['float64', 'int64']): Selects columns from the dataframe df3 that have numeric data types (float64 and int64).

numeric\_cols1.corr(): Computes the Pearson correlation coefficients between all pairs of numeric columns.

Values range from -1 to 1:

* **1: Perfect positive correlation.**
* **0: No correlation.**
* **-1: Perfect negative correlation.**

**annot**=True: Displays correlation values directly on the heatmap.

**fmt=".**2f": Formats the correlation values to two decimal places.

**cmap**="coolwarm": Sets the color map to "coolwarm," where blue represents negative correlations and red represents positive correlations.

**Purpose of Correlation Analysis:**

1-Identifies linear relationships between variables.

2-Highlights variables that are strongly correlated (positively or negatively), which can:

Indicate redundancy (e.g., features with near-perfect correlations).

Reveal potential predictors or interactions.

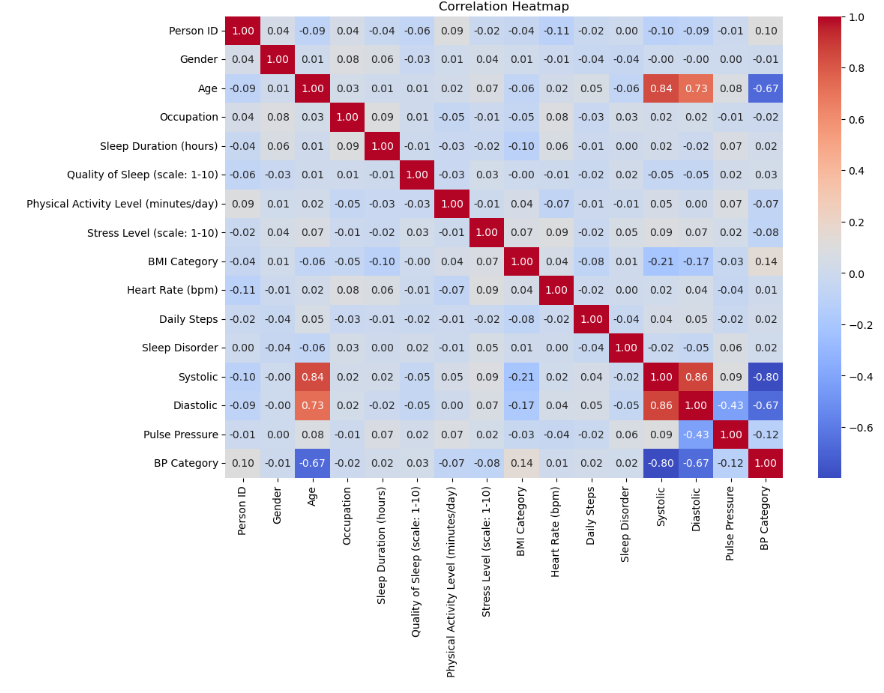
Interpretation of the Heatmap:

**Diagonal Elements:** Always 1.0 because each variable is perfectly correlated with itself.

**Strong Positive Correlation** (near 1): Indicates variables move in the same direction (e.g., "Daily Steps" and "Physical Activity Level").

**Strong Negative Correlation** (near -1): Indicates variables move in opposite directions (e.g., "Stress Level" and "Quality of Sleep").

**Weak/No Correlation** (near 0): Suggests no linear relationship between variables.



then we apply a scatter plot for the relation between variables:

A white background with text

AI-generated content may be incorrect.

A graph of a diagram

AI-generated content may be incorrect.

sns.scatterplot: Generates a scatter plot for the specified x-axis (Systolic) and y-axis (Age).

data=df3: Uses the dataframe df3 as the source of data.

plt.title("Systolic vs. Age"): Adds a title to describe the plot.

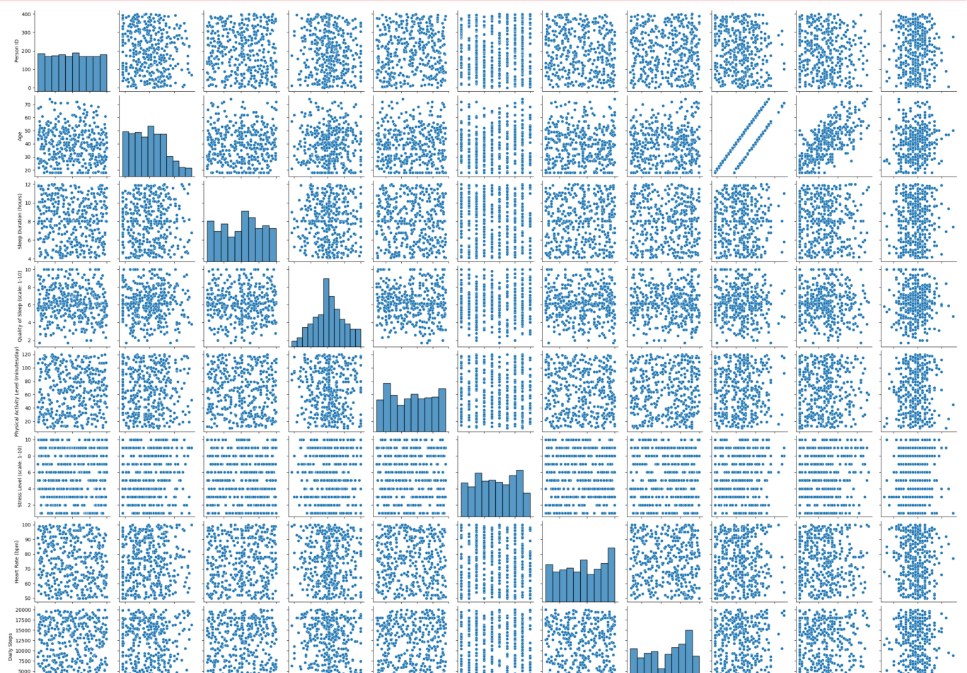
**Purpose:**

Examines the specific relationship between Systolic Blood Pressure and Age.

Highlights trends, correlations, or clusters within the data.

A close-up of a computer screen

AI-generated content may be incorrect.



sns.pairplot: Creates a grid of scatter plots for all pairwise combinations of numeric columns.

numeric\_cols: The selected numeric columns are used for plotting.

**Purpose:**

Provides an overview of relationships between variables.

Highlights patterns such as linear trends, clusters, or outliers.

Includes histograms along the diagonal to show the distribution of individual variables.

**A close-up of a computer screen

AI-generated content may be incorrect.Aggregation and filtering :**

The code performs data aggregation and filtering to compute the average sleep duration for individuals, grouped by their gender. Aggregating data by gender helps identify patterns in sleep behavior between groups.

**Group by Gender:**

df1.groupby('Gender'):

The dataset (df1) is grouped based on the unique values in the "Gender" column.

This means the data is divided into groups, such as "Male" and "Female."

**Aggregation:**

.agg({'Sleep Duration (hours)': 'mean'}):

The aggregation function computes the mean (average) of the "Sleep Duration (hours)" column for each gender group. {} is used to specify that aggregation is performed only on this specific column, and the operation (mean) is explicitly defined.

**Resetting Index:**

.reset\_index():

Converts the grouped data from a hierarchical index back into a flat DataFrame structure, making it easier to read and work with.

**Age Filtering and Stress Levels:** **Filtering patients older than 50 and calculating the average stress level grouped by blood pressure (BP) category.**

Older patients (over 50) show varying stress levels depending on their BP category, which may inform health recommendations.

Occupation-Based Patterns:

A screenshot of a computer code

AI-generated content may be incorrect.Occupatios influence sleep, stress, and activity levels. For instance, retired individuals might have higher average sleep duration and lower stress compared to manual laborers.

**Aggregating Multiple Metrics:**

.agg({...}):

Specifies the columns and aggregation operations:

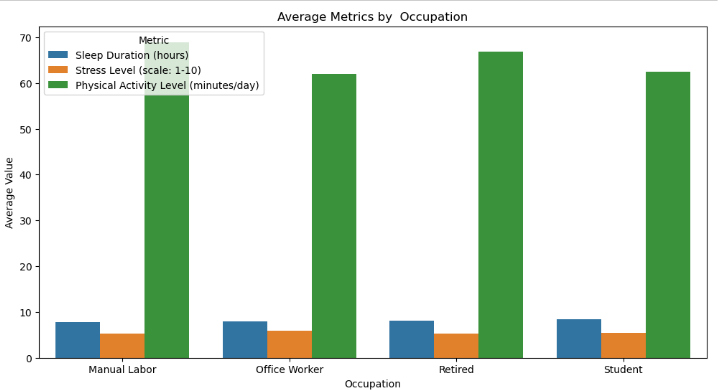
"Sleep Duration (hours)": "mean": Computes the average sleep duration for each occupation.

"Stress Level (scale: 1-10)": "mean": Computes the average stress level for each occupation.

"Physical Activity Level (minutes/day)": "mean": Computes the average physical activity level for each occupation.

A screenshot of a computer code

AI-generated content may be incorrect.



**Visualization of Aggregated Data (Bar Plot)**

Visualize the average sleep duration, stress level, and physical activity grouped by occupation. This Shows how key metrics vary by occupation, helping identify trends such as higher stress levels or lower physical activity in certain jobs.

**Melt the Data:**

agg\_data.melt(id\_vars=["Occupation"], var\_name="Metric", value\_name="Average Value"):

Reshapes the data into a long format, creating a column called "Metric" (e.g., Sleep Duration, Stress Level) and their corresponding values under "Average Value." var\_name="Metric": Creates a column called metric which contains the names of the aggregated metrics ( Sleep Duration (hours), Stress Level (scale: 1-10)).

**sns.barplot(...):**

Plots a bar graph showing the average value of metrics for each occupation. Different metrics are distinguished by the hue="Metric" parameter.

**Distribution of Sleep Duration Across Age Groups (Box Plot)**

Analyze how sleep duration varies across different age groups. Highlights trends like shorter sleep durations in younger or older age groups.

**Create Age Groups**:

pd.cut(df3["Age"], bins=[0, 20, 40, 60, 80], labels=["0-20", "20-40", "40-60", "60-80"]):

Divides the "Age" column into intervals (bins) with labels for easy categorization.

**Plot Sleep Duration:**

sns.boxplot(...):

Box plots show the distribution (median, quartiles, outliers) of sleep duration for each age group. Sleep Duration Across Age Groups

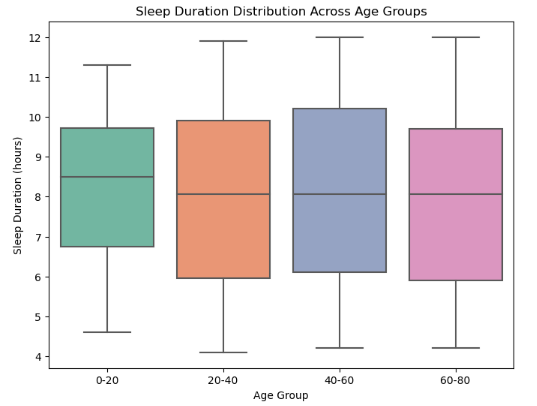
The goal is to see how Sleep Duration (hours) varies across Age Groups.

A box plot is ideal because:

It shows the median sleep duration for each age group.

The interquartile range (IQR) reveals the variability in sleep durations.

Outliers (if any) are clearly displayed, indicating unusual sleep patterns.



A computer screen shot of a computer code

AI-generated content may be incorrect.

**Distribution of Stress Levels Across Age Groups (Box Plot)**

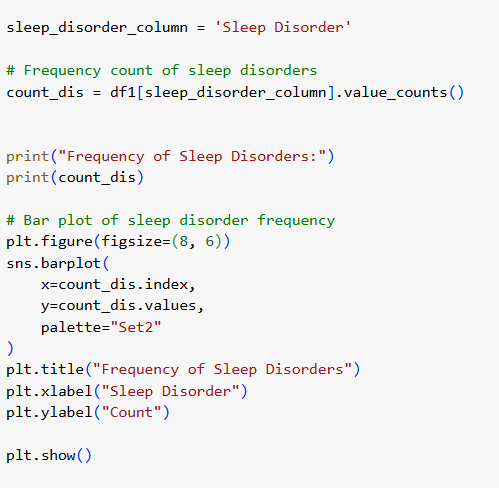
Visualize stress level distribution across finer age intervals. Explores how stress varies with age, potentially correlating with life stages.

**Define Finer Age Groups:**

Divides ages into smaller intervals for detailed analysis: [0, 20, 30, 40, 50, 60, 70, 80].

**Plot Stress Levels:**

Box plots display stress level variability within each age group.

**Frequency of Sleep Disorders (Bar Plot)**

Visualize the frequency of various sleep disorders in the dataset. Provides an overview of the prevalence of sleep disorders in the dataset.

**Count Sleep Disorders:**

df1[sleep\_disorder\_column].value\_counts():

Counts occurrences of each unique value in the "Sleep Disorder" column.

**Plot Frequency:**

**sns.barplot(...):**

Displays a bar graph showing the count of each sleep disorder type.

**Why we used boxplot:**

A box plot (also known as a box-and-whisker plot) is used in the code for analyzing the distribution of **Sleep Duration** and **Stress Levels** across different **Age Groups**.

**Group Comparisons:**

Box plots allow easy comparison of multiple groups (e.g., age groups) side by side.

**Compact Representation:**

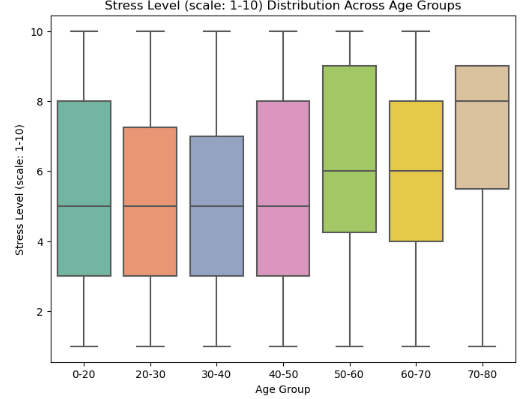
A lot of information (median, quartiles, range, and outliers) is packed into a single visualization.

**Highlighting Outliers:**

Outliers are crucial for understanding extreme cases, which might need special attention (e.g., very low or very high stress/sleep levels).

**Detecting Patterns:**

Patterns like shorter sleep durations or higher stress levels in specific age ranges can be easily identified.



**Sleep Duration Across Age Groups**

The goal is to see how Sleep Duration (hours) varies across Age Groups.

A box plot is ideal because:

It shows the median sleep duration for each age group.

The interquartile range (IQR) reveals the variability in sleep durations.

Outliers (if any) are clearly displayed, indicating unusual sleep patterns.

**Why we used barplot :**

Reason:

A bar plot is ideal for comparing quantitative data (average values) across categories (occupations).The hue parameter adds another layer, showing multiple metrics side-by-side for each occupation.

A bar plot effectively conveys the mean values of metrics across occupations, making it easy to analyze and present trends.It allows the audience to quickly identify insights such as:

Which occupation has the lowest physical activity?

How stress levels vary by occupation compared to sleep duration.

A graph with a number of different colored squares

AI-generated content may be incorrect.



**Average Metrics by Sleep Disorder Status**

Calculate and visualize the average metrics for individuals based on their sleep disorder status.

**Group Data by Sleep Disorder:**

df1.groupby("Sleep Disorder"):

Groups the dataset by the unique values in the "Sleep Disorder" column.

**Aggregate Metrics:**

**.agg({...}):**

Computes the mean for multiple columns like: Sleep Duration, Quality of Sleep, StressLevel, Physical Activity Level, Heart Rate , Daily Steps ,Age

**.reset\_index():**

Converts the grouped data back to a flat DataFrame for easier manipulation and visualization.

**Visualize Average Age:**

**plt.barh(...):**

Creates a horizontal bar chart to show the average age for each sleep disorder category.

**Visualization Insights:**

Horizontal bar charts are simple and effective for comparing average age across sleep disorder statuses.Helps identify age groups more prone to specific sleep disorders

A screenshot of a computer program

AI-generated content may be incorrect.Aggregating Blood Pressure Metrics by BP Category

Code :

Summarize systolic and diastolic blood pressure values (min, max, mean) for each blood pressure (BP) category.

Group Data by BP Category:

df1.groupby("BP Category"):

Groups the dataset by categories like "Normal," "Elevated," or "Hypertension."

Aggregate Metrics:

.agg({...}):

Computes:

Minimum (min), maximum (max), and mean (mean) values for systolic and diastolic blood pressure.

Reset Index:

.reset\_index():

Returns the grouped data as a flat DataFrame.

Results:

Displays the summarized BP metrics for each category.

**Visualization Insights:**

Highlights the variability of BP levels in different BP categories.

Normalization :

Using ColumnTransformer and Pipeline



Purpose:

To scale selected numerical columns to a uniform range of [0, 1] using MinMaxScaler, ensuring consistent feature magnitudes across the dataset.

We apply normalization to the following numeric features:

* **Daily Steps**
* **Sleep Duration (hours)**
* **Physical Activity Level (minutes/day)**
* **Heart Rate (bpm)**

These features are chosen because they vary in scale and need to be brought to a common range for better model performance.

**Using Pipeline and ColumnTransformer**

1. **Pipeline Setup**  
   A Pipeline is created that applies MinMaxScaler to the numeric features
2. **ColumnTransformer**  
   A ColumnTransformer is used to apply the scaler only to the selected numeric features, while leaving the rest of the columns (if any) unchanged
3. **Applying the Transformation**  
   The transformation is applied to the original DataFrame (df1)
4. **Creating the Transformed DataFrame**  
   The result is converted back into a DataFrame with updated column names

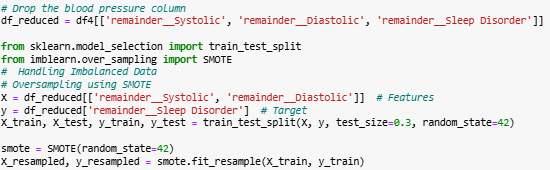
**Why Normalize?**

* Ensures that features with large numeric ranges (e.g., steps in thousands) don't dominate smaller ones (e.g., heart rate in tens).
* Helps machine learning models (especially gradient-based ones) converge faster and perform better.

**Saving the Result**

The normalized data is saved as a JSON file:

transformed\_df.to\_json(r"C:\Users\Nagham\Downloads\sleepdata\_normalized.json", orient='records', lines=False)



**Dropping Unnecessary Columns**

Remove the Blood Pressure (systolic/diastolic) column as it may not be needed or relevant for the current analysis.

**Handling Imbalanced Data Using SMOTE.**

**Problem:**

The Sleep Disorder column (target variable) is imbalanced, meaning some categories occur much more frequently than others, which can bias the model.

**Solution:**

Use SMOTE (Synthetic Minority Oversampling Technique) to balance the classes by oversampling the minority class.

**Train-Test Split:**

train\_test\_split(...) splits data into training and testing sets.

test\_size=0.3 reserves 30% of the data for testing.

**Apply SMOTE:**

smote.fit\_resample(X\_train, y\_train):

Generates synthetic samples for the minority class in the training set.

Balances the training set for better model performance.

**Check Class Distribution:**

y\_resampled.value\_counts():

Verifies the balanced class distribution after SMOTE.

**Why SMOTE?**

Prevents model bias toward majority classes.

Improves model performance on underrepresented classes.

we made oversampling because the data is imbalanced , as you saw before the distribution of the sleep disorder class is not distributed well , the "No" class has the greatest frequancy,that it makes a huge difference between it and other classes

**Visualizing Class Distribution After SMOTE**

**Bar Plot:**

**sns.barplot(...):**

Plots the frequency of each class in the resampled training set.

x=new\_class\_dist.index and y=new\_class\_dist.values specify the labels and values for the classes.

**Customize Plot:**

Titles and labels are added to enhance readability.

**Purpose:**

Demonstrates the effectiveness of SMOTE in balancing the dataset.



created to ensure dependency isolation and avoid conflicts between packages. These specific parts of the project were implemented using **PyCharm**, a professional integrated development environment (IDE) for Python. Unlike the rest of the code, which was developed and executed in **Jupyter Notebook**, PyCharm was used for these scripts due to its better handling of file structures, external libraries (such as pyquery), and multiprocessing operations. This approach ensured clean execution and better management of the modules involved.

Multiprocessing is imported to enable parallel processing, which helps speed up tasks by running them on multiple CPU cores simultaneously.

**Reading CSV and Converting to JSON:**

data = pd.read\_csv(r"C:\Users\Nagham\Downloads\Datascience Programming\proj\new.csv")

data.to\_json(r"C:\Users\Nagham\Downloads\Datascience Programming\proj\original\_data.json")

pd.read\_csv(...) reads the CSV file located at the specified path into a pandas DataFrame called data.

data.to\_json(...) converts and saves the DataFrame into a JSON file, making the data easier to work with in later stages.

**Defining a Function to Read JSON:**

def read\_sleep\_json(filename):

return pd.read\_json(filename)

This function takes a filename as input and reads the JSON file into a pandas DataFrame.

It modularizes the JSON reading process so it can be used multiple times, especially in multiprocessing.

**Using Multiprocessing to Read Multiple JSON Files Concurrently:**

if \_\_name\_\_ == "\_\_main\_\_":

file1 = r"C:\Users\Nagham\Downloads\Datascience Programming\proj\original\_data.json"

file2 = r"C:\Users\Nagham\Downloads\Datascience Programming\proj\resampled\_sleep\_data.json"

pool = multiprocessing.Pool(processes=2)

df1\_result = pool.apply\_async(read\_sleep\_json, args=(file1,))

df2\_result = pool.apply\_async(read\_sleep\_json, args=(file2,))

pool.close()

pool.join()

The if \_\_name\_\_ == "\_\_main\_\_": guard ensures that the multiprocessing code runs only when this script is executed directly.

multiprocessing.Pool(processes=2) creates a pool with 2 worker processes to run tasks in parallel.

apply\_async asynchronously runs the read\_sleep\_json function on both JSON files at the same time.

pool.close() prevents any more tasks from being submitted to the pool.

pool.join() waits for all the worker processes to finish.

**Combining the Results:**

combined\_df = pd.concat([df1\_result.get(), df2\_result.get()], axis=0, ignore\_index=True)

print("✅ Combined data shape:", combined\_df.shape)

print(combined\_df.head())

df1\_result.get() and df2\_result.get() retrieve the DataFrames returned by the asynchronous function calls.

pd.concat(...) merges the two DataFrames vertically (stacking rows).

ignore\_index=True resets the index of the combined DataFrame.

Finally, the shape and first 5 rows of the combined DataFrame are printed for verification.

**Result Explanation:**

✅ Combined data shape: (999, 17)

Person ID Gender Age ... Diastolic Pulse Pressure BP Category

0 1.0 Male 29.0 ... 70.0 54.0 Elevated

1 2.0 Female 43.0 ... 86.0 45.0 Elevated

2 3.0 Male 44.0 ... 70.0 52.0 Elevated

3 4.0 Male 29.0 ... 72.0 52.0 Elevated

4 5.0 Male 67.0 ... 78.0 55.0 Elevated

The combined dataset has 999 rows and 17 columns, indicating the successful merging of both JSON datasets.

The displayed first five rows show sample data columns like Person ID, Gender, Age, and blood pressure-related measurements.

This confirms that the multiprocessing approach worked correctly, and the data is now ready for further analysis.

**Why this approach?**

Using multiprocessing speeds up reading large JSON files by loading them concurrently instead of sequentially.

Converting from CSV to JSON allows flexibility in handling the data, as JSON is widely used in data exchange and APIs.

Combining datasets prepares a unified dataset for analysis without manual intervention.

Web Scraping Sleep Tips Using PyQuery and Requests

* requests: Used to send HTTP requests and fetch content from websites.
* pyquery: A library similar to jQuery but for Python, used to parse and extract elements from HTML documents.
* json: Used to save the extracted data in a structured JSON format.

**Scraping Function:**

def scrape\_sleep\_tips\_pyquery():

url = "https://www.sleepfoundation.org/sleep-hygiene"

print(f"📡 Fetching content from: {url}")

* Defines a function named scrape\_sleep\_tips\_pyquery.
* The url variable holds the target webpage for scraping sleep hygiene tips.
* A message is printed to indicate the beginning of the web request.

**Adding Headers to Mimic a Browser:**

headers = {

"User-Agent": (

"Mozilla/5.0 (Windows NT 10.0; Win64; x64) "

"AppleWebKit/537.36 (KHTML, like Gecko) "

"Chrome/114.0.0.0 Safari/537.36"

)

}

* Websites often block automated requests. This custom User-Agent header makes the request appear as if it came from a real web browser like Chrome.

**Sending the Request and Parsing Content:**

try:

response = requests.get(url, headers=headers)

if response.status\_code == 200:

doc = pq(response.text)

tips = []

* The try block handles potential errors during the request.
* requests.get(...) fetches the web page content.
* A successful response (status\_code == 200) means the page loaded properly.
* pq(response.text) parses the HTML content of the page.
* An empty list tips is created to store the extracted tips.

Extracting Sleep Tips from Lists:

for li in doc("ul li, ol li").items():

tip = li.text()

if tip:

tips.append(tip)

* This loop selects all list items (li) inside both unordered (ul) and ordered (ol) lists on the page.
* .text() extracts only the visible text of each list item.
* Non-empty tips are added to the tips list.

**Saving Extracted Tips to a JSON File:**

with open("sleep\_tips\_pyquery.json", "w", encoding="utf-8") as f:

json.dump(tips[:10], f, indent=2, ensure\_ascii=False)

* Only the first 10 tips are saved to a JSON file called sleep\_tips\_pyquery.json.
* ensure\_ascii=False allows for proper saving of any non-ASCII characters (e.g., accents).
* indent=2 formats the file for better readability.

**Preview and Completion Message:**

print("✅ Sleep tips saved to sleep\_tips\_pyquery.json")

for i, tip in enumerate(tips[:5], 1):

print(f"{i}. {tip}")

* Confirmation message is shown after saving.
* First 5 tips are printed to the console as a quick preview for the user.

**Handling Errors and Invalid Requests:**

else:

print("❌ Failed to fetch page, status code:", response.status\_code)

except Exception as e:

print("❌ Error scraping tips:", e)

* If the response is not successful, a failure message with the status code is displayed.
* If an exception (e.g., network error or parsing issue) occurs, it is caught and printed.

**Run the Function Only When Script Is Executed Directly:**

if \_\_name\_\_ == "\_\_main\_\_":

scrape\_sleep\_tips\_pyquery()

* This condition ensures the function runs only if the script is executed directly (not imported as a module).

**Result Example:**

📡 Fetching content from: https://www.sleepfoundation.org/sleep-hygiene

✅ Sleep tips saved to sleep\_tips\_pyquery.json

1. Stick to a sleep schedule

2. Pay attention to what you eat and drink

3. Create a restful environment

4. Limit daytime naps

5. Include physical activity in your daily routine

**Why Use Web Scraping?**

* Purpose: Automatically collect sleep hygiene tips from a reliable medical source.
* Advantage: No need to manually copy content — the program extracts and saves tips for further use in reports, recommendation systems, or visualizations.
* Efficiency: Using pyquery makes it easy to select HTML elements, and requests ensures smooth content access.

***Conclusion:***

In this project, a comprehensive approach to data engineering and analysis was employed to uncover meaningful insights from a dataset focused on sleep health and lifestyle factors. The process began with systematic data cleaning and preprocessing, which involved handling missing values, removing duplicates, addressing outliers using the IQR method, and transforming raw features—such as splitting the blood pressure string into numerical systolic and diastolic values. Categorical variables, including gender, occupation, and sleep disorders, were encoded numerically to enable effective analysis. Additionally, numerical features were normalized to ensure consistency and comparability across variables, and irrelevant columns were removed to focus on those most relevant to the study’s objectives. Exploratory Data Analysis (EDA) followed, using aggregation techniques and grouped analyses to identify patterns across demographic and health-related variables. Relationships between sleep duration, stress levels, physical activity, and sleep quality were explored across different groups categorized by gender, age, occupation, and sleep disorder status. Visualization tools such as bar plots and box plots were utilized to communicate these findings clearly, revealing important trends and differences in health behaviors and outcomes. To ensure the reliability of predictive modeling, the class imbalance in the target variable (sleep disorder) was addressed through synthetic oversampling using the SMOTE technique. This step balanced the dataset, reducing bias toward majority classes and allowing for fairer, more accurate modeling, especially for underrepresented groups. Overall, the combination of thorough data preprocessing, insightful EDA, and targeted balancing techniques demonstrates the critical role of data engineering in transforming raw datasets into actionable insights. The visualizations and findings not only validate the robustness of the methodology but also lay a solid foundation for future predictive modeling, emphasizing the potential of data-driven approaches in improving our understanding of sleep health and its relationship with lifestyle factors.

A close-up of a computer screen

AI-generated content may be incorrect.A white background with black text

AI-generated content may be incorrect.