SLEEP METRICS INSIGHTS: EXPLORING SLEEP HEALTH PATTERNS THROUGH DATA MINING AND ANALYTICS

Iteration 4 BDAS

Github: https://github.com/JinkyXiang/INFOSYS722-BDAS

Jingqi Xiang (Jinky Xiang)
Jxia436@aucklanduni.ac.nz

Table of Contents

1. Business and/or Situation understanding

1.1. Identify the objectives of the business/situation

Did you get enough sleep last night? Do you sometimes feel tired even after sleeping for over 9 hours? When was the last time you woke up feeling refreshed? On average, humans spend one-third of their lives sleeping, which translates to 15 years of sleep for a person who is 75 years old. Quality sleep plays a crucial role in restoring and repairing physiological and psychological processes in our bodies. Unfortunately, insufficient sleep has become a prevalent issue in modern society, with around 37% of New Zealanders sleeping less than 7 hours, while the optimal sleeping duration is 7-9 hours (Lee & Sibley, December 2019). Sleep disorders, such as insomnia, can have adverse effects on overall health and well-being (Vgontzas & Fernandez-Mendoza, 2013).

The primary objective of the project is to address sleeping-related issues by leveraging data mining and big data analytics techniques with a dataset of sleep duration and lifestyle as presented in Figure 1. A study will be conducted with the following objectives:

- Exploring the association between demographic factors, lifestyle habits, sleep patterns, and overall sleep health. Specifically, it will investigate how different sleep patterns, including sleep duration and quality, and lifestyle factors, such as physical activity levels and stress, relate to overall sleep health.
- Developing robust predictive models capable of accurately identifying sleep disorders in patients.
- Providing applicable recommendations for sleep disorder issues. Develop personalised recommendations for individuals experiencing sleep-related issues, such as sleep disorders. These recommendations will encompass factors based on data analysis to help improve sleep health.

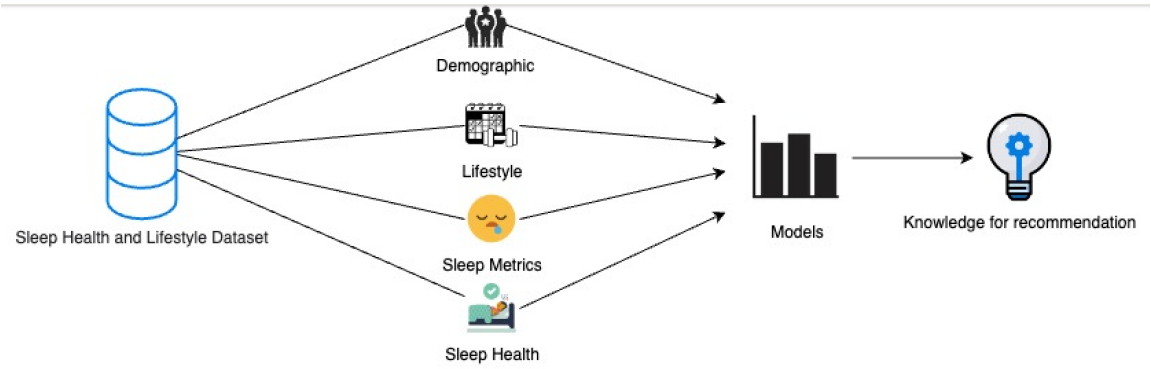

Figure 1: Initial presentation of how the study may be conducted

The study will be considered a success if:

- The study reveals meaningful associations and relationships between sleep disorders, sleep metrics and lifestyle
- Achieve a minimum prediction accuracy rate of 85% for sleep disorder detection within the developed predictive models.
- Although it is difficult to implement personalised recommendations for patients in this study, positive feedback from users who receive personalised recommendations would further validate its success.

1.2. Assess the situation

1.2.1. Recourses

a) Dataset: The Sleep Health and Lifestyle Dataset from Kaggle.com, comprising 375 data entries with 13 variables encompassing sleep metrics, physical and mental health indexes, and lifestyle factors.

b) Analysing tool: Python will be used as the primary analysis tool for this project. As a high-level general-purpose programming language, it offers a comprehensive set of tools for data mining, including data processing, modelling, and analysis.

c) Personnel: Myself as the data analyst, and tutors as database experts who may also be consulted

1.2.2. Requirements

a) Data security and legal restrictions: The data must be handled securely under legal regulation to protect privacy of the individuals.

b) Project scheduling: To meet the deadline set by course INFOSYS 722 at the University of Auckland, efficient project scheduling and time management are imperative.

1.2.3. Assumptions

a) Representative dataset: The Sleep Health and Lifestyle Dataset is assumed to be representative of the targeted population's sleep-related issues and lifestyle factors, with minimal impact from external factors or biases.

b) Data quality: The dataset is assumed to have sufficient quality for data mining and big data analysis.

1.2.4. Constrains

a) Data availability limitation: The absence of specific sleep quality-related data, such as sleep latency, sleep waking, wakefulness, and sleep efficiency, may constrain the depth of analysis. The subjective number provided for sleep quality in the dataset may be a constraint due to the lack of objective data

b) Data size limitation: With only 375 data entries, the relatively small dataset size may limit the complexity and granularity of data mining analyses

1.2.5. Risks

a) Scheduling: The project may encounter scheduling challenges if the data analyst is occupied with other ongoing projects. Delays in project completion could impact meeting the specified deadline.

b) Insufficient data analysis: The data mining may not yield sufficient or significant insights, particularly if the data size is relatively small, which may lead to an inability to implement data interpretation.

c) Technology problems: Technical issues with the Spyder and Anaconda software could disrupt the data mining and analysis processes.

1.2.6. Contingencies

a) Scheduling: Implement a detailed project plan with clear milestones to ensure the project is completed step by step to meet the deadline. Prioritise tasks to minimise conflicts with other projects.

b) Insufficient data analysis: Additional sources may require if the analysis yield is insufficient

c) Technology problems: Identify potential software providers or experts who can offer support and troubleshooting

1.3. Determine data mining objectives

The goal of data mining is to:

- Use feature selection method to identify the most influential factors related to sleep health
- Perform analysis to explore relationships between sleep health and the influential factors
- Develop predictive models that can forecast sleep-related issues based on various factors, such as an individual's demographic and lifestyle
- Use clustering algorithms to group individuals with similar sleep patterns and habits

The success criteria for data mining are:

- Identify the most importance factors related to sleep health by ranking
- Discovers robust correlation between sleep health metrics and importance factors.
- Provides effective clustering results that successfully group individuals with similar sleep patterns
- Achieve high accuracy in the performance of the predictive model.

1.4. Produce a project plan

The project outline is presented in the table below:

Phase	Time	Resources	Risks
Situation understanding	23 Sep – 24 Sep 2023	All resources	personnel misunderstanding the situation
Data understanding	23 Sep – 26 Sep 2023	All resources	data unavailability
Data preparation	27 Sep – 30 Sep 2023	All resources	Technology problems
Data transformation	1 Oct – 2 Oct 2023	All resources	Insufficient data analysis, Technology problems
Data-mining method(s) selection	3 Oct – 4 Oct 2023	All resources	Insufficient data analysis, Technology problems
Data-mining algorithm(s) selection	5 Oct – 7 Oct 2023	All resources	Insufficient data analysis, Technology problems
Data Mining	8 Oct – 10 Oct 2023	All resources	Insufficient data analysis, Technology problems
Interpretation	11 Oct – 12 Oct 2023	All resources	Inability to implement data interpretation

2. Data understanding

2.1. Collect initial data

The initial data for the project was collected from the Sleep Health and Lifestyle Dataset, which was obtained from kaggle.com, a reputable open-source dataset website. Kaggle is a massive community for data science that facilitates dataset discovery and publication, as well as global collaboration among data scientists. The dataset can be accessed from https://www.kaggle.com/datasets/uom190346a/sleep-health-and-lifestyle-dataset. However, the collaborator of the dataset did not specify the collection method used to gather information. I chose this dataset for my project for the following reasons:

- The dataset is relevant to healthcare and well-being, which is one of the 17 sustainable development goals.
- The dataset is provided in CSV format, which can be easily imported and analysed using Python in Spyder software, ensuring seamless data processing.
- The dataset exhibits good diversity in terms of variables related to sleep and daily habits. This helps to capture different patterns and avoid bias from homogenous datasets.

However, a significant challenge I encountered during data collection was striking the right balance between dataset volume and computational limitations. A large dataset could slow down my laptop, while a small one may not be sufficient for full data mining. To optimise the dataset's size and ensure accurate analysis, I have to use careful consideration.

2.2. Describe the data

I will be using Python through Spyder software to process the dataset provided in a Comma Separated Values (CSV) format. The dataset contains 375 rows (data entries) and 13 columns (fields). The dataset includes characteristics related to sleep health which is relevant to the issue address in the business understanding phase. It contains information such as sleep duration, sleep quality, physical activity level, blood pressure and presence of sleep disorder. All the attributes are important factors that could impact sleep health.

The dataset contains both symbolic and numeric data. Here is the breakdown of each field:

1. Personal ID: Unique identifier for each individual.
2. Gender: Categorise individuals as male or female can provide insights into potential gender-based differences in sleep health.
3. Age: Age of each individual in years.
4. Occupation: Description of each person's occupation or profession.
5. Sleep Duration: Number of hours each individual sleeps per day.
6. Quality of Sleep: Subjective rating of sleep quality on a scale of 1 to 10 (bad sleep to good sleep).
7. Physical Activity Level: Number of minutes of physical activity engaged in per day.
8. Stress Level: Subjective rating of stress levels experienced on a scale of 1 to 10 (not stressed to very stressed).
9. BMI Category: Categorise of individuals as underweight, normal, overweight, or obese.
10. Blood Pressure: Indication of systolic/diastolic values.
11. Heart Rate: Resting heart rate in beats per minute(bpm).
12. Daily Steps: Number of steps taken by each individual per day.
13. Sleep Disorders: Presence or absence of sleep disorders, including ‘None’, ‘Insomnia’, and ‘Sleep Apnea’.

The Figure 2 shows detailed information for the first 5 data entries from the dataset. And Figure 3 shows the column names and the dataset of 375 rows and 13 columns.

```
# Display up to 5 rows at once
df.show(n=5)
```

```
+---------+------+---+--------------------+--------------+----------------+------------------------+------------+-----
-------+--------------+----------+-----------+--------------+
|Person ID|Gender|Age|          Occupation|Sleep Duration|Quality of Sleep|Physical Activity Level|Stress Level|BMI C
ategory|Blood Pressure|Heart Rate|Daily Steps|Sleep Disorder|
+---------+------+---+--------------------+--------------+----------------+------------------------+------------+-----
-------+--------------+----------+-----------+--------------+
|        1|  Male| 27|   Software Engineer|           6.1|               6|                      42|           6|  Ove
rweight|        126/83|        77|       4200|          None|
|        2|  Male| 28|              Doctor|           6.2|               6|                      60|           8|
Normal|        125/80|        75|      10000|          None|
|        3|  Male| 28|              Doctor|           6.2|               6|                      60|           8|
Normal|        125/80|        75|      10000|          None|
|        4|  Male| 28|Sales Representative|           5.9|               4|                      30|           8|
Obese|        140/90|        85|       3000|   Sleep Apnea|
|        5|  Male| 28|Sales Representative|           5.9|               4|                      30|           8|
Obese|        140/90|        85|       3000|   Sleep Apnea|
+---------+------+---+--------------------+--------------+----------------+------------------------+------------+-----
-------+--------------+----------+-----------+--------------+
only showing top 5 rows
```

Figure 2: First five data information in Sleep Health and Lifestyle Dataset

```
#columns details
df.columns
```

```
['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']
```

```
#coutn columns
column_count = len(df.columns)
"Number of columns:", column_count
```

```
('Number of columns:', 13)
```

```
#count rows
row_count = df.count()
"Number of rows:", row_count
```

```
('Number of rows:', 375)
```

Figure 3: Details of columns and rows

Below are some basic statistics for key attributes of the dataset:

Figure 4 displays general statistics of the DataFrame. However, the result appears messy due to the presence of too many columns. To enhance readability, I have separated the columns and created three tables, as shown in Figure 5. 'Gender', 'Occupation', 'BMI Category', 'Blood Pressure', and 'Sleep Disorder' are not numeric values, so their mean and standard deviation are not calculated. Most columns have a count of 375, except for 'Blood Pressure', 'Heart Rate', and 'Daily Steps', which have a count of 374. This suggests that there is one missing value in these columns.

```
#general statistics
df.describe().show()
```

```
+-------+------------------+------+-----------------+----------+------------------+------------------+---------------
--------+------------------+-----------+--------------+-----------------+-------------+
|summary|         Person ID|Gender|              Age|Occupation|    Sleep Duration|  Quality of Sleep|Physical Activi
ty Level|      Stress Level|BMI Category|Blood Pressure|       Heart Rate|       Daily Steps|Sleep Disorder|
+-------+------------------+------+-----------------+----------+------------------+------------------+---------------
--------+------------------+-----------+--------------+-----------------+-------------+
|  count|               375|   375|              375|       375|               375|               375|
375|               375|        375|           374|              374|              374|           375|
|   mean|             188.0|  null|           42.136|      null| 7.134933333333324| 7.314666666666667|      59.226666
66666667| 5.381333333333333|       null|          null|70.16577540106952| 6816.844919786096|          null|
| stddev|108.39741694339399|  null|8.712285259247578|      null|0.7965036789098551|1.1958812262546714|      20.830724
25730269|1.7735951986568852|       null|          null|4.135675535112212|1617.9156791336366|          null|
|    min|                 1|Female|               24|Accountant|               5.8|                 4|
30|                 3|      Normal|        115/75|               65|             10000|      Insomnia|
|    max|                99|  Male|               59|   Teacher|               8.5|                 9|
90|                 8|  Overweight|        142/92|               86|              8000|   Sleep Apnea|
+-------+------------------+------+-----------------+----------+------------------+------------------+---------------
--------+------------------+-----------+--------------+-----------------+-------------+
```

Figure 4: General statistics of dataset

```
In [48]: # Select the first 6 columns by index
selected_columns = df.columns[:6]

# Create a new DataFrame with the selected columns
selected_df = df.select(selected_columns)

# Calculate and display summary statistics for the selected columns
selected_describe = selected_df.describe()
selected_describe.show()

# Select and create 6-10 columns by index
selected_columns2 = df.columns[6:10]
selected_df2 = df.select(selected_columns2)

# display
selected_df2.describe().show()

# Select and create remainingn columns
selected_columns3 = df.columns[10:]
selected_df3 = df.select(selected_columns3)

# display
selected_df3.describe().show()
```

```
+-------+------------------+------+-----------------+----------+------------------+------------------+
|summary|         Person ID|Gender|              Age|Occupation|    Sleep Duration|  Quality of Sleep|
+-------+------------------+------+-----------------+----------+------------------+------------------+
|  count|               375|   375|              375|       375|               375|               375|
|   mean|             188.0|  null|           42.136|      null| 7.134933333333324| 7.314666666666667|
| stddev|108.39741694339399|  null|8.712285259247578|      null|0.7965036789098551|1.1958812262546714|
|    min|                 1|Female|               24|Accountant|               5.8|                 4|
|    max|                99|  Male|               59|   Teacher|               8.5|                 9|
+-------+------------------+------+-----------------+----------+------------------+------------------+

+-------+----------------------+-----------------+-----------+--------------+
|summary|Physical Activity Level|      Stress Level|BMI Category|Blood Pressure|
+-------+----------------------+-----------------+-----------+--------------+
|  count|                   375|              375|        375|           374|
|   mean|      59.22666666666667|5.381333333333333|       null|          null|
| stddev|      20.83072425730269|1.7735951986568852|       null|          null|
|    min|                    30|                3|     Normal|        115/75|
|    max|                    90|                8| Overweight|        142/92|
+-------+----------------------+-----------------+-----------+--------------+

+-------+-----------------+------------------+--------------+
|summary|       Heart Rate|       Daily Steps|Sleep Disorder|
+-------+-----------------+------------------+--------------+
|  count|              374|               374|           375|
|   mean|70.16577540106952| 6816.844919786096|          null|
| stddev|4.135675535112212|1617.9156791336366|          null|
|    min|               65|             10000|      Insomnia|
|    max|               86|              8000|   Sleep Apnea|
+-------+-----------------+------------------+--------------+
```

Figure 5: Separated tables of general statistics

The Figure 6 displays the data type of the variables using the 'print schema' command. However, all of the data seems to be of type String, which is not appropriate. For instance, age and quality of sleep should be of type integer, while sleep duration should be of type double. The manipulation of data types will be performed in the later section.

```
: df.printSchema()
```

```
root
 |-- Person ID: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: string (nullable = true)
 |-- Occupation: string (nullable = true)
 |-- Sleep Duration: string (nullable = true)
 |-- Quality of Sleep: string (nullable = true)
 |-- Physical Activity Level: string (nullable = true)
 |-- Stress Level: string (nullable = true)
 |-- BMI Category: string (nullable = true)
 |-- Blood Pressure: string (nullable = true)
 |-- Heart Rate: string (nullable = true)
 |-- Daily Steps: string (nullable = true)
 |-- Sleep Disorder: string (nullable = true)
```

Figure 6: Data type of variables

2.3. Explore the data

Exploration1:

Visualisation is primarily used to understand the surface levels and patterns of a dataset. However, interpreting data through visual representations or tables can be challenging when the data is in string format. To address this issue, I first convert variables with numerical data from string to numeric format, as demonstrated in the Figure 7. Person ID, Age, Quality of Sleep, Physical Activity Level, Blood Pressure, Heart Rate, and Daily Steps are converted to integers as they do not possess decimal places. Sleep Duration is transformed into a floating-point number, as it includes decimal values.

```
#import in the relevant types.
from pyspark.sql.functions import col
from pyspark.sql.types import (StructField,StringType,IntegerType,StructType,FloatType)

# Define the new data types
new_data_types = [
    StructField('Person ID', IntegerType(), True),
    StructField('Gender', StringType(), True),
    StructField('Age', IntegerType(), True),
    StructField('Occupation', StringType(), True),
    StructField('Sleep Duration', FloatType(), True),
    StructField('Quality of Sleep', IntegerType(), True),
    StructField('Physical Activity Level', IntegerType(), True),
    StructField('Stress Level', IntegerType(), True),
    StructField('BMI Category', StringType(), True),
    StructField('Blood Pressure', IntegerType(), True),
    StructField('Heart Rate', IntegerType(), True),
    StructField('Daily Steps', IntegerType(), True),
    StructField('Sleep Disorder', StringType(), True)
]

# Apply the new data types to the DataFrame
for field in new_data_types:
    df = df.withColumn(field.name, col(field.name).cast(field.dataType))

df.printSchema()
```

```
root
 |-- Person ID: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Occupation: string (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Quality of Sleep: integer (nullable = true)
 |-- Physical Activity Level: integer (nullable = true)
 |-- Stress Level: integer (nullable = true)
 |-- BMI Category: string (nullable = true)
 |-- Blood Pressure: integer (nullable = true)
 |-- Heart Rate: integer (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Disorder: string (nullable = true)
```

Figure 7: Converting data type to numeric

Figure 8 shows the correlation values between all the numeric variables. A value closer to one indicates a stronger positive correlation, while a value closer to negative one indicates a stronger negative correlation. The results indicate a strong positive correlation between sleep duration and quality of sleep, as well as between Daily Steps and Physical Activity Level. In contrast, there is a strong negative correlation between Sleep Duration and Stress Level, as well as between Quality of Sleep and Stress Level.

```
Correlation between Quality of Sleep vs Sleep Duration: 0.8904392669405827
Correlation between Sleep Duration vs Quality of Sleep: 0.8904392669405825
Correlation between Stress Level vs Quality of Sleep: -0.80013090900132
Correlation between Quality of Sleep vs Stress Level: -0.8001309090013199
Correlation between Daily Steps vs Physical Activity Level: 0.7492723044584496
Correlation between Physical Activity Level vs Daily Steps: 0.7492723044584495
Correlation between Sleep Duration vs Stress Level: -0.6623911537605
Correlation between Stress Level vs Sleep Duration: -0.6623911537604998
Correlation between Stress Level vs Heart Rate: 0.5224135422696888
Correlation between Heart Rate vs Stress Level: 0.5224135422696888
Correlation between Age vs Quality of Sleep: 0.5055505982294969
Correlation between Quality of Sleep vs Age: 0.5055505982294968
Correlation between Age vs Sleep Duration: 0.3962011565261073
Correlation between Sleep Duration vs Age: 0.3962011565261073
Correlation between Stress Level vs Age: -0.3604596215943737
Correlation between Age vs Stress Level: -0.36045962159437367
Correlation between Physical Activity Level vs Sleep Duration: 0.25418114315740803
Correlation between Sleep Duration vs Physical Activity Level: 0.254181143157408
Correlation between Quality of Sleep vs Heart Rate: -0.2450903037835158
Correlation between Heart Rate vs Quality of Sleep: -0.2450903037835158
Correlation between Quality of Sleep vs Physical Activity Level: 0.226863827462983
Correlation between Physical Activity Level vs Quality of Sleep: 0.226863827462983
Correlation between Stress Level vs Daily Steps: 0.2167143842722831
Correlation between Daily Steps vs Stress Level: 0.2167143842722831
Correlation between Daily Steps vs Heart Rate: 0.21066245824231344
Correlation between Heart Rate vs Daily Steps: 0.2106624582423134
Correlation between Age vs Physical Activity Level: 0.2004448350450795
Correlation between Physical Activity Level vs Age: 0.20044483504507948
Correlation between Heart Rate vs Physical Activity Level: 0.13667701552029426
Correlation between Physical Activity Level vs Heart Rate: 0.13667701552029415
Correlation between Daily Steps vs Age: 0.12554268502797986
Correlation between Age vs Daily Steps: 0.12554268502797977
Correlation between Heart Rate vs Sleep Duration: -0.09759698068661661
Correlation between Sleep Duration vs Heart Rate: -0.09759698068661651
Correlation between Quality of Sleep vs Daily Steps: 0.07195584087996929
Correlation between Daily Steps vs Quality of Sleep: 0.07195584087996924
Correlation between Age vs Heart Rate: 0.05410113367784663
Correlation between Heart Rate vs Age: 0.05410113367784659
Correlation between Daily Steps vs Sleep Duration: 0.03989560047244017
Correlation between Sleep Duration vs Daily Steps: 0.03989560047244011
Correlation between Physical Activity Level vs Stress Level: -0.012842515232345716
Correlation between Stress Level vs Physical Activity Level: -0.012842515232345638
```

Figure 8: Correlation between numeric variables

Exploration2:

A histogram of sleep duration was generated using the matplotlib library. The figure is shown in Figure 9. The results showed that a considerable number of individuals in the population under study are sleeping below the recommended optimal hours (7 hours), suggesting a prevalent issue of insufficient sleep.

To gain a better understanding of the sleep duration of individuals with optimal sleep hours, I generated a table using an SQL query. The table counts individuals with sleep duration greater than or equal to 7 hours and less than 7 hours, as shown in Figure 10. Out of a total of 375 individuals, 155 sleep less than 7 hours, accounting for approximately 41.33% of the sample. This represents a substantial number of individuals who do not achieve optimal sleep on a daily basis.

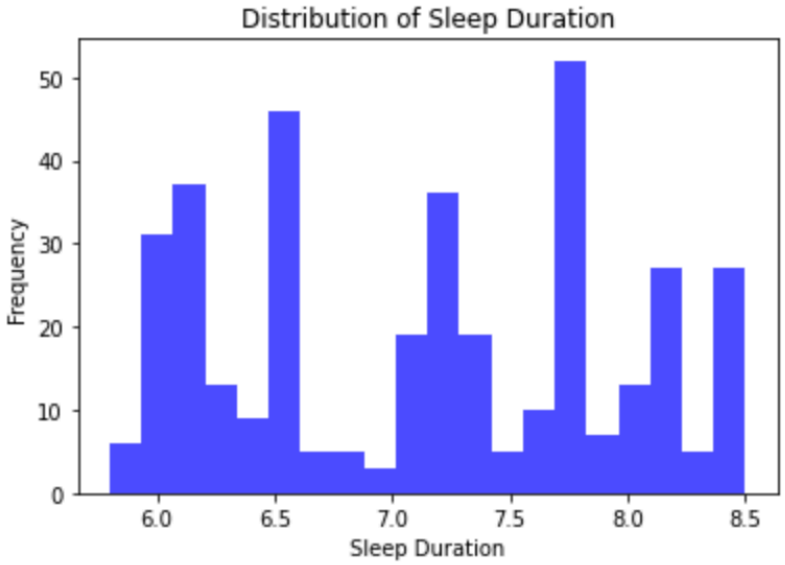

Figure 9: Histogram of sleep duration of all individuals

```
#register your DataFrame as a temporary SQL table
df.createOrReplaceTempView("sleep_table")

#SQL query to get counts for Sleep Duration < 7 and >= 7
sleep_duration_distribution = spark.sql("""
    SELECT
        CASE WHEN CAST(`Sleep Duration` AS FLOAT) < 7 THEN '< 7 hours'
             WHEN CAST(`Sleep Duration` AS FLOAT) >= 7 THEN '>= 7 hours'
             ELSE 'Unknown'
        END AS SleepDurationCategory,
        COUNT(*) AS Count
    FROM sleep_table
    WHERE `Sleep Duration` IS NOT NULL
    GROUP BY SleepDurationCategory
""")
#show the Sleep Duration distribution
sleep_duration_distribution.show()
```

```
+--------------------+-----+
|SleepDurationCategory|Count|
+--------------------+-----+
|           < 7 hours|  155|
|          >= 7 hours|  220|
+--------------------+-----+
```

Figure 10: Sleep duration greater and equal than 7 hours and less than 7 hours

Exploration3:

The value count (Figure 11) and bar chart and of genders(Figure 12) in the dataset indicates a close balance between male and female individuals. This balanced representation reduces the likelihood of gender bias, enhancing the dataset's reliability.

```
#3. Values in Gender (use SQL)
#register your DataFrame as a temporary SQL table
df.createOrReplaceTempView("sleep_table")

#use SQL queries to get gender counts
gender_distribution = spark.sql("SELECT Gender, COUNT(*) AS Count FROM sleep_table WHERE Gender != 'Gender' GROUP
gender_distribution.show()
```

```
+------+-----+
|Gender|Count|
+------+-----+
|Female|  186|
|  Male|  189|
+------+-----+
```

Figure 11: Gender counts

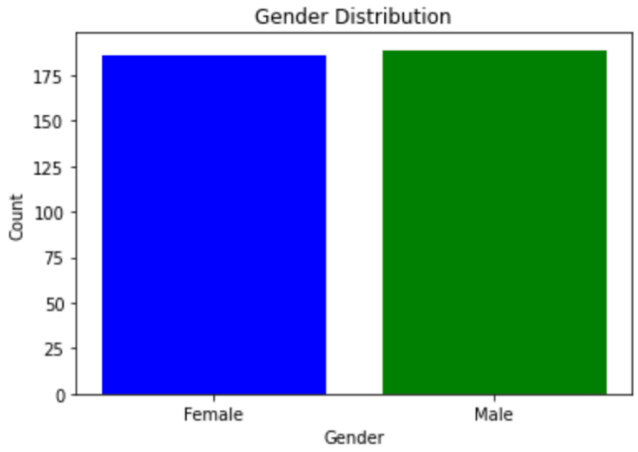

Figure 12: Bar chart and value counts of gender

Exploration4:

Figure 13 shows the histogram of sleep quality, while Figure 14 displays the exact number of individuals with different sleep quality. Sleep quality ratings of 6 and 8 have the highest frequencies, indicating that the majority of individuals have moderate to high sleep quality. This suggests that most people are satisfied with their sleep. However, there are some individuals who report poor sleep quality, which warrants further investigation into factors that may affect sleep quality.

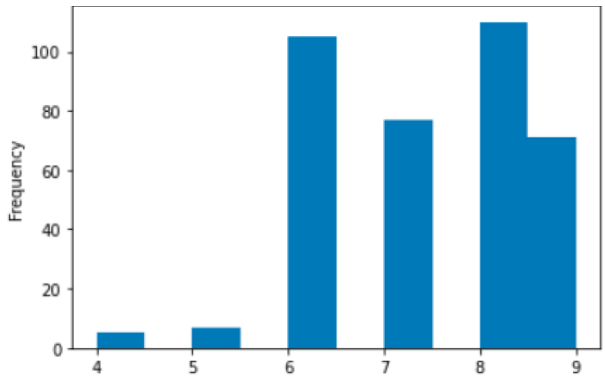

Figure 13: Distribution of sleep quality

```
#4. Distribution of Quality of Sleep
df.groupBy('Quality of Sleep').count().orderBy(col('Quality of Sleep')).show()
```

```
+----------------+-----+
|Quality of Sleep|count|
+----------------+-----+
|               4|    5|
|               5|    7|
|               6|  105|
|               7|   77|
|               8|  110|
|               9|   71|
+----------------+-----+
```

Figure 14: Value count for Quality of sleep

Exploration5:

The bar chart of sleep disorder is shown in Figure 15. The chart shows a notable number of individuals are experiencing sleep disorders, like insomnia and sleep apnea. It's important to address these issues among the population in the study. More analysis is needed to find connections between sleep disorders and other sleep health metrics and lifestyle factors. Understanding the impact of these sleep disorders can contribute to enhancing overall sleep health and well-being.

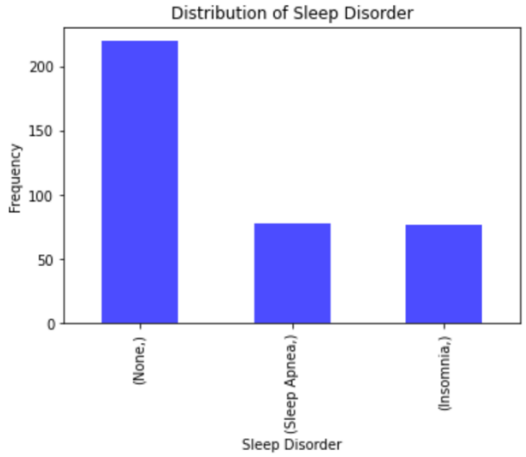

Figure 15: Bar chart of people in sleep disorder

Exploration5:

As shown in Figure 16, sleep duration is strongly correlated with sleep quality. Longer sleep hours are expected to result in higher sleep quality, while shorter sleep duration may lead to lower sleep quality.

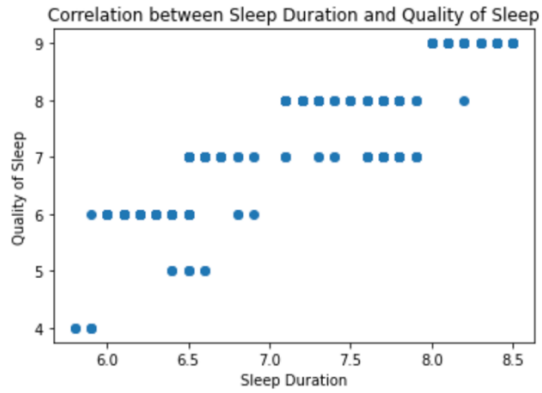

Figure 16: Scatterplot of sleep quality and sleep duration

Exploration6:

There might be a trend suggesting that as age increases, sleep quality improves. However, as shown in Figure 17, age correlation with sleep quality might not be significant. To further understand the pattern of age and sleep quality, I used SQL query to calculate average quality of sleep for each age group as shown in Figure 18. I then converted the results into line plots, as shown in Figure 19. Individuals over the age of 55 have a very high average sleep quality. Overall, there does not seem to be a clear pattern between age and quality of sleep.

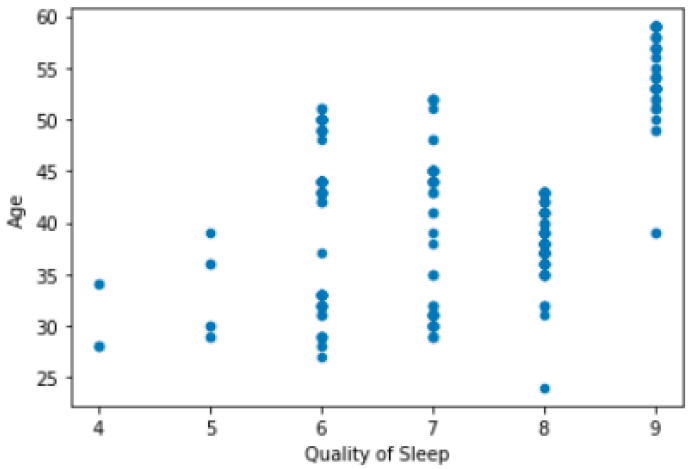

Figure 17: Scatterplot of sleep quality and age

```
#register the DataFrame as a temporary table
df.createOrReplaceTempView("sleep_data")

from pyspark.sql.functions import avg, round

#calculate the average of Quality of Sleep grouped by Age and round to 2 decimal places
average_quality_of_sleep = df.groupBy('Age').agg(round(avg('Quality of Sleep'), 2).alias('Average Quality of Sleep')).c

#show all the result
average_quality_of_sleep.show(n=df.count(), truncate=False)
```

```
+---+------------------------+
|Age|Average Quality of Sleep|
+---+------------------------+
|24 |8.0                     |
|27 |6.0                     |
|28 |4.8                     |
|29 |6.15                    |
|30 |6.69                    |
|31 |6.89                    |
|32 |6.53                    |
|33 |6.0                     |
|34 |4.0                     |
|35 |7.75                    |
|36 |7.5                     |
|37 |7.9                     |
|38 |7.95                    |
|39 |7.87                    |
|40 |8.0                     |
|41 |7.83                    |
|42 |7.33                    |
|43 |7.09                    |
|44 |6.4                     |
|45 |7.0                     |
|48 |6.67                    |
|49 |6.55                    |
|50 |6.15                    |
|51 |7.63                    |
|52 |7.67                    |
|53 |9.0                     |
|54 |9.0                     |
|55 |9.0                     |
|56 |9.0                     |
|57 |9.0                     |
|58 |9.0                     |
|59 |9.0                     |
+---+------------------------+
```

Figure 18: Age and average quality of sleep

```
#extract the Age and Average Quality of Sleep columns
age = average_quality_of_sleep.select('Age').toPandas()['Age']
quality_of_sleep = average_quality_of_sleep.select('Average Quality of Sleep').toPandas()['Average Quality of Sleep']

#plot the Age and Average Quality of Sleep
plt.plot(age, quality_of_sleep, marker='o')
plt.title('Average Quality of Sleep by Age')
plt.xlabel('Age')
plt.ylabel('Average Quality of Sleep')
plt.show()
```

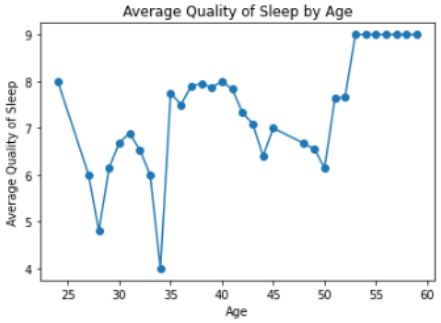

Figure 19: Plot of average quality of sleep and age

Exploration7:

Figure 20 shows the correlation between Physical Activity Level, Quality of Sleep, and Stress Level. As the x-axis moves to the right, the dots move up and gradually change colour from yellow to purple. This suggests a relationship between physical activity level, stress levels, and sleep quality. Lower stress levels and higher physical activity are likely to result in higher sleep quality, while higher stress and less physical activity can lead to poorer sleep quality.

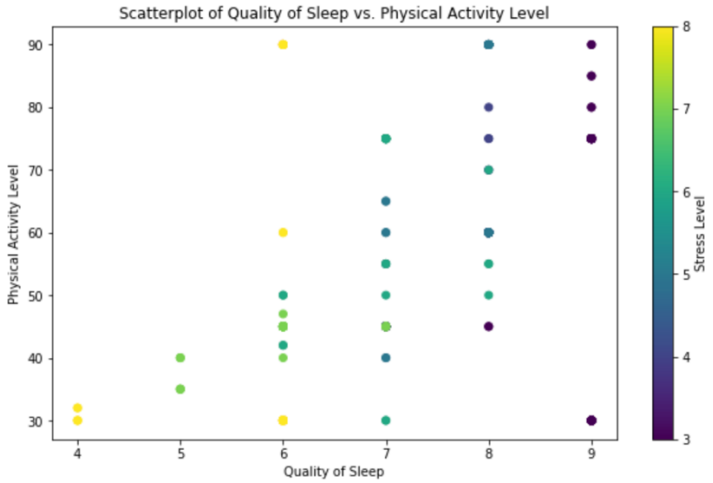

Figure 20: Scatter plot of quality of sleep, physical activity and stress level

Exploration8:

Low sleep quality could potentially contribute to the development of sleep disorders. However, as demonstrated in Figure 21, individuals who do not have sleep disorders all had above-average sleep quality, while individuals with sleep disorders (sleep apnea and isomnia) have varying levels of sleep quality. To gain further insights into the relationship between sleep quality and various sleep health patterns, I categorized all individuals based on their Sleep Disorder status and calculated the average sleep quality for each group, as presented in Figure 22. Interestingly, there is not a significant disparity in average sleep quality among the three sleep health groups—'None,' 'Sleep Apnea,' and 'Insomnia,' with values of 7.63, 7.21, and 6.53, respectively. This suggests that there may be other contributing factors influencing the occurrence of sleep disorders.

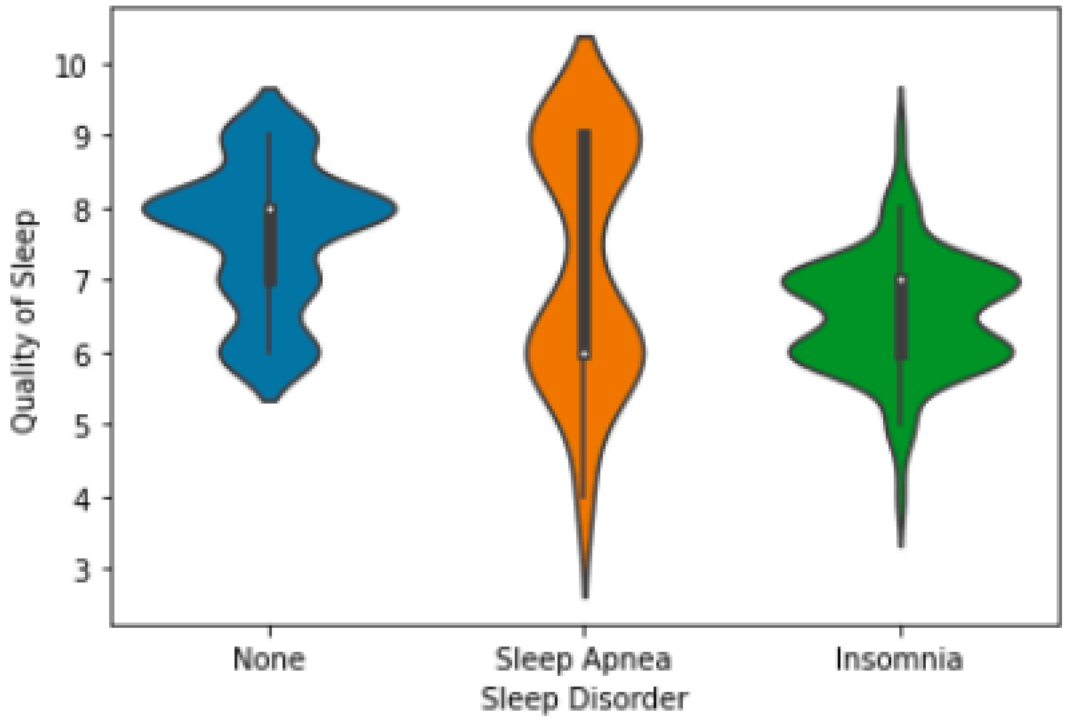

Figure 21: Violin plot of sleep disorder and sleep quality

```
from pyspark.sql.functions import avg

# Calculate the average of Quality of Sleep grouped by Sleep Disorder
average_quality_of_sleep = df.groupBy('Sleep Disorder').agg(avg('Quality of Sleep').alias('Average Quality of Sle

# Show the result
average_quality_of_sleep.show()
```

```
+--------------+------------------------+
|Sleep Disorder|Average Quality of Sleep|
+--------------+------------------------+
|          None|       7.627272727272727|
|   Sleep Apnea|       7.205128205128205|
|      Insomnia|       6.532467532467533|
+--------------+------------------------+
```

Figure 22: Sleep disorder and average quality of sleep

Promising Attributes for Further Analysis:

Sleep duration and sleep quality: Investigating their correlation and patterns. If a strong correlation is found, this study can aid individuals struggling with low sleep quality by providing evidence that longer sleep duration may improve sleep quality. If not, further exploration can provide valuable insights into distinct aspects of sleep health, allowing for more precise analysis when investigating the correlation of sleep patterns with other factors.

Sleep disorder and sleep pattern: The initial data exploration provides a picture of sleep disorders in dataset and potential factors that may influence sleep patterns. By exploring how sleep disorders and patterns are correlated, we can better understand how to reduce their impact on sleep health. We can introduce more factors like gender and BMI measurement in to the study. This understanding can help us identify factors that contribute to sleep disturbances, and develop personalized interventions for individuals with sleep-related issues.

As for identify particular subsets of data for later use, it is crucially important to identify the most influential factors related to sleep health. This information can help us find effective interventions for sleep-related issues. In terms of data mining goals, exploring the relationship between sleep patterns, lifestyle, demographic factors, and sleep health can lead to the development of a predictive model that can forecast and discover potential sleep disorders. Currently, important parameters like sleep disorder and gender are categorised as 'object' data type, which does not provide much insight in data analysis. Converting these data types to numeric or binary types may offer more insight when analysing associations between factors. Therefore, choosing appropriate data types for parameters should be considered important in order to achieve data mining goals.

2.4. Verify the data quality

Overall, the data quality is good, with no measurement errors, coding inconsistencies, or bad metadata. However, there are a few incomplete fields shown in Figure 23. Specifically, 'Heart Rate', and 'Daily Steps' each have one missing value. These can be processed in the data preparation step.

There is a coding inconsistencies concerning the BMI measurement, as shown in Figure 24. The BMI category in the dataset includes values such as 'normal', 'normal weight', 'overweight', and 'obese'. To ensure consistency, 'normal' and 'normal weight' should be categorized as the same measure. This requires further action during the data preparation stage to address the BMI category discrepancy and ensure accurate analysis.

```
#check null values
#iterate through each column and count the number of null values
for col in df.columns:
    null_count = df.filter(df[col].isNull()).count()
    print(f"Column '{col}': {null_count} null values")
```

```
Column 'Person ID': 0 null values
Column 'Gender': 0 null values
Column 'Age': 0 null values
Column 'Occupation': 0 null values
Column 'Sleep Duration': 0 null values
Column 'Quality of Sleep': 0 null values
Column 'Physical Activity Level': 0 null values
Column 'Stress Level': 0 null values
Column 'BMI Category': 0 null values
Column 'Blood Pressure': 0 null values
Column 'Heart Rate': 1 null values
Column 'Daily Steps': 1 null values
Column 'Sleep Disorder': 0 null values
```

Figure 23: Sum of null values of each column

```
#check detail of 'BMI Category'
#group the data by 'BMI Category' and count the occurrences of each category
bmi_category_counts = df.groupBy('BMI Category').count().orderBy('count', ascending=False)

#show the details
bmi_category_counts.show()
```

```
+-------------+-----+
| BMI Category|count|
+-------------+-----+
|       Normal|  196|
|   Overweight|  148|
|Normal Weight|   21|
|        Obese|   10|
+-------------+-----+
```

Figure 24: Distribution of BMI

3. Data preparation

3.1. Select the data

The dataset contains a wide range of variables related to sleep and daily routines. This stage involves making thoughtful decisions about which data to include and exclude. The dataset should be aligned with business goals and data mining objectives prior to any analytical procedures.

Selecting Items: All the 375 individuals' data are selected for initial study. Because each individual's dataset contributes valuable and informative data to the study.
Selecting Attributes: The primary aim of this study is to uncover relations among lifestyle, sleep patterns, and sleep health. To align our analysis closely with our study objectives and ensure its efficiency, we have chosen to filter attributes that are not directly pertinent to these goals. In iterations 2 and 3, I initially removed Blood Pressure and Heart Rate from the dataset. However, after conducting a literature review, I discovered that some studies have found a link between poor sleep patterns and an increased risk of hypertension (Li & Shang, Aug 2021), . As a result, I have decided to keep Blood Pressure in the data frame.
Regarding Heart Rate, according to the study 'Dynamics of Heart Rate and Sleep Stages in Normals and Patients with Sleep Apnea', the sleeping heart rate decreases with each apnea due to cardiovascular changes (Penzel, Kantelhardt , Lo , Voigt, & Vogelmeier , 2003). However, the heart rate data in the dataset represents resting heart rate, which is measured when individuals are awake and completely at rest. In contrast, sleeping heart rate is measured while individuals are asleep and can vary throughout the different phases of the sleep cycle. Therefore, when comparing sleeping heart rate and resting heart rate, I can conclude that resting heart rate lacks a direct relationship with sleep patterns and sleep health. As a result, we will omit resting heart rate from the analysis. This streamlined approach allows us to focus on the most relevant factors.

To exclude the 'Heart Rate' variable, run the code as shown in Figure 25. After executing the code, the data information displays a decrease in the number of fields from 13 to 12 and 'Heart Rate' is removed from the dataset, indicating that the specified attributes have been successfully excluded as shown in Figure 26.

```
#create a new DataFrame 'df_select' by dropping the 'Heart Rate' column
df_select = df.drop('Heart Rate')

#show the 'df_select' DataFrame with first 2 rows
df_select.show(2)
```

```
+---------+------+---+-----------------+--------------+----------------+----------------------+------------+--------
---+--------------+-----------+--------------+
|Person ID|Gender|Age|       Occupation|Sleep Duration|Quality of Sleep|Physical Activity Level|Stress Level|BMI Cate
gory|Blood Pressure|Daily Steps|Sleep Disorder|
+---------+------+---+-----------------+--------------+----------------+----------------------+------------+--------
---+--------------+-----------+--------------+
|        1|  Male| 27|Software Engineer|           6.1|               6|                    42|           6|  Overwe
ight|        126/83|       4200|          None|
|        2|  Male| 28|           Doctor|           6.2|               6|                    60|           8|       No
rmal|        125/80|      10000|          None|
+---------+------+---+-----------------+--------------+----------------+----------------------+------------+--------
---+--------------+-----------+--------------+
only showing top 2 rows
```

Figure 25: Excluding blood pressure and heart rate

```
#show variables
print("columns:", df_select.columns)

#coutn columns
column_count = len(df_select.columns)
print("Number of columns:", column_count)
```

```
columns: ['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration', 'Quality of Sleep', 'Physical Activity Leve
l', 'Stress Level', 'BMI Category', 'Blood Pressure', 'Daily Steps', 'Sleep Disorder']
Number of columns: 12
```

Figure 26: Fields reduce from 13 to 12

3.2. Clean the data

The original dataset from Kaggle was very clean and did not contain any significant dirt as shown in Figure 27. All the columns has 374 count values which means there is not null value. To practice data cleaning skills in using python, I manually added a dirty data point. This was the individual data for ID 375, which had daily steps, blood pressure and heart rate are set to null or none as shown in Figure 28. In the following paragraphs, I will treat the added data as if it were originally present in the dataset.

```
#import original dataset
df_original = spark.read.csv('Datasets/Sleep_health_and_lifestyle_dataset(original).csv', inferSchema=True, header=True
```

```
#the count of values in original dataset
for col in df_original.columns:
    count = df_original.select(col).count()
    print(f"Count of values in column '{col}': {count}")
```

```
Count of values in column 'Person ID': 374
Count of values in column 'Gender': 374
Count of values in column 'Age': 374
Count of values in column 'Occupation': 374
Count of values in column 'Sleep Duration': 374
Count of values in column 'Quality of Sleep': 374
Count of values in column 'Physical Activity Level': 374
Count of values in column 'Stress Level': 374
Count of values in column 'BMI Category': 374
Count of values in column 'Blood Pressure': 374
Count of values in column 'Heart Rate': 374
Count of values in column 'Daily Steps': 374
Count of values in column 'Sleep Disorder': 374
```

Figure 27: Original dataset from Kaggle (no added dirt)

```
# Show the last row in the DataFrame
# Collect all rows into a list
all_rows = df.collect()

# Get the last row (which is the last element in the list)
last_row = all_rows[-1]

print(last_row)
```

```
Row(Person ID=375, Gender='Female', Age=24, Occupation='Teacher', Sleep Duration=8.199999809265137, Quality of Sleep=
8, Physical Activity Level=80, Stress Level=4, BMI Category='Normal', Blood Pressure='125/80', Heart Rate=None, Daily
Steps=None, Sleep Disorder='None')
```

Figure 28: The dirt data added to the dataset

Missing Data: Figure 23 shows that there is one null value in daily steps, as indicated by the null summary figure. This null value is due to the individual not completing the questionnaire, which may potentially introducing noise into subsequent models. Since the individual's personal contact information is not available from the dataset publisher, it is impossible to ask them to complete the questionnaire. To remove the noise, I imputed the missing value using the fixed method, setting it to the mean of daily steps in the dataset. The Histogram of daily steps shown in Figure 29, resembles a bell-shaped curve, indicating that daily steps among individuals are randomly distributed. Therefore using the mean value to replace the null value has minimal impact on the relationships between variables, while preserving the overall distribution of data.

The coding of imputing missing values is presented in Figure 30. First, calculate the mean of the daily steps. Then, replace any null values in 'Daily Steps' with the calculated mean value. The figure also shows the data information after executing the code, where 'null' values have been replaced by mean value '6816', resulting in no missing values in 'Daily Steps' now.

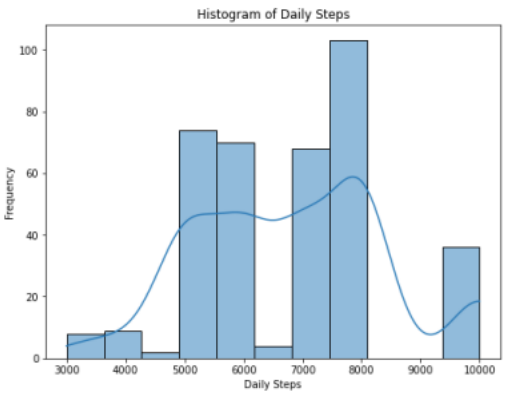

Figure 29: Histogram of daily steps among individuals

```
from pyspark.sql.functions import mean

# Calculate the mean of the 'Daily Steps' column
mean_daily_steps = df_select.select(mean("Daily Steps")).collect()[0][0]

# Fill missing values in 'Daily Steps' with the mean value
df_select = df_select.na.fill(mean_daily_steps, subset=['Daily Steps'])

#show the 375th row
row_with_id_375 = df_select.filter(df_select["Person ID"] == 375)
row_with_id_375.show()
```

```
+---------+------+---+----------+--------------+----------------+----------------------+------------+------------+--
-----------+-----------+--------------+
|Person ID|Gender|Age|Occupation|Sleep Duration|Quality of Sleep|Physical Activity Level|Stress Level|BMI Category|Bl
ood Pressure|Daily Steps|Sleep Disorder|
+---------+------+---+----------+--------------+----------------+----------------------+------------+------------+--
-----------+-----------+--------------+
|      375|Female| 24|   Teacher|           8.2|               8|                    80|           4|      Normal|
125/80|     6816|          None|
+---------+------+---+----------+--------------+----------------+----------------------+------------+------------+--
-----------+-----------+--------------+
```

Figure 30: Imputing missing value in daily step

Coding Inconsistencies: As shown in Figure 24, the BMI input value includes both 'normal' and 'normal weight', which have the same meaning. If the algorithm treats them as different inputs, it may lead to inaccurate insights. Therefore, I grouped 'Normal' and 'Normal weight' into 'Normal' weight, as shown in Figure 31, to ensure that only one concept is counted for data analysis.

Figure 31 displays the grouping code. The code combines instances of 'Normal weight' and 'Normal'. I used 'lower' and 'trim' function to convert parameter names to lower case and remove whitespace. The resulting output shows three unique 'BMI Category' values: 'Normal', 'Overweight', and 'Obese'. The value for 'Normal' has increased.

```
from pyspark.sql.functions import when, col, trim, lower

# Create a new DataFrame 'df_clean' with the updated 'BMI Category'
df_clean = df_select.withColumn(
    "BMI Category",
    when((lower(trim(df_select["BMI Category"])) == "normal weight") |
         (lower(trim(df_select["BMI Category"])) == "normal"), "Normal")
    .otherwise(df_select["BMI Category"])
)

# Group the data by 'BMI Category' and count the occurrences of each category
bmi_category_counts = df_clean.groupBy('BMI Category').count().orderBy('count', ascending=False)

# Show the details
bmi_category_counts.show()
```

```
+-------------+-----+
|BMI Category|count|
+-------------+-----+
|       Normal|  217|
|  Overweight|  148|
|        Obese|   10|
+-------------+-----+
```

Figure 31: BMI information after grouping 'Normal' and 'Normal Weight'

Overall, two data problems were cleaned up to improve data quality, and there were no cases or attributes that could not be salvaged.

3.3. Construct the data

In line with the study's objective of exploring the relationships between sleep patterns, lifestyle, and sleep disorders, while also predicting potential sleep disorder occurrences and offering recommendations, two additional attributes could be introduced.

First attribute:

The first attribute is to change column name ‘Sleep Disorder’ to ‘Sleep Health’ and create a new field called ‘Sleep Disorder’ with binary data to indicate if the individual has sleep disorder. This is to give better distinguish and understanding between individuals with normal sleep and with sleep issues. It employs a binary data structure that indicates the ‘Presence’ (true) or ‘Absence’ (false) of a sleep disorder for each individual. This field unifies 'Insomnia' and 'Sleep Apnea' under the same group, effectively grouping distinct sleep disorders. The binary classification aligns with the study objective and enhances interpretability and suitability for modelling techniques. Categorising individuals into two groups allows for easier comparison and seamless communication, benefiting both researchers and stakeholders.

The coding for renaming is shown in Figure 32, the column 'Sleep Disorder' is renamed to 'Sleep Pattern', as it contains original data about individuals' sleep patterns. As shown in Figure 33, a new column is created based on 'Sleep Pattern'. Individuals with 'Insomnia' or 'Sleep Apnea' are labelled as 'true', indicating that they have a sleep disorder, while those with 'None' are labelled as 'false', indicating that they do not have a sleep disorder. The figure also displays the data frame information after generating the sleep disorder column. The Sleep Disorder column is now Boolean type and has 220 of true values and 155 of false counts.

```
df_construct = df_select.withColumnRenamed("Sleep Disorder", "Sleep Health")

df_construct.printSchema()
```

```
root
 |-- Person ID: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Occupation: string (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Quality of Sleep: integer (nullable = true)
 |-- Physical Activity Level: integer (nullable = true)
 |-- Stress Level: integer (nullable = true)
 |-- BMI Category: string (nullable = true)
 |-- Blood Pressure: string (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Health: string (nullable = true)
```

Figure 32: Renaming field of Sleep Health

```
df_construct = df_construct.withColumn(
    "Sleep Disorder",
    when((df_construct["Sleep Health"] == "Insomnia") | (df_construct["Sleep Health"] == "Sleep Apnea"), True)
    .otherwise(False)
)

#show result
df_construct.printSchema()
Sleep_Disorder_counts = df_construct.groupBy('Sleep Disorder').count().orderBy('count', ascending=False)
Sleep_Disorder_counts.show()
```

```
root
 |-- Person ID: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Occupation: string (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Quality of Sleep: integer (nullable = true)
 |-- Physical Activity Level: integer (nullable = true)
 |-- Stress Level: integer (nullable = true)
 |-- BMI Category: string (nullable = true)
 |-- Blood Pressure: string (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Health: string (nullable = true)
 |-- Sleep Disorder: boolean (nullable = false)
```

```
+--------------+-----+
|Sleep Disorder|count|
+--------------+-----+
|         false|  220|
|          true|  155|
+--------------+-----+
```

Figure 33: Sleep Disorder column

Second attributes:

The second addition is a new column called ‘BMI Status’. This attribute also uses a Boolean data format to indicate an individual's BMI measurement from BMI Category as either 'True' (normal) or ‘False’ (abnormal). Within this classification, individuals classified as 'Overweight' and 'Obese' are consolidated abnormal and under the 'False' category, as both denote higher body fatness compared to 'Normal'. The binary classification effectively groups distinct individual’s BMI status. And this enables a clearer exploration of patterns and trends related to sleep disorders across different BMI categories.

The code shown in Figure 34. It is similar to the code for 'Sleep Health'. A new column called ‘BMI Status’ is created from the ‘BMI Category’ column. Individuals with a ‘Normal’ BMI are classified as 'true', while those with any other BMI category are classified as 'false'. The value 'Normal' is assigned to 'true', indicating that the individual has a normal BMI. The value 'Abnormal' is assigned to 'false', indicating that the individual has a BMI that is overweight or obese. The figure also displays the data frame information after generating the BMI Status column. The BMI Status column is now Boolean type and has 217 of true values and 158 of false counts.

```
# create new column call BMI Status
df_construct2 = df_construct.withColumn(
    "BMI Status",
    when((df_construct["BMI Category"] == "Normal"), True)
    .otherwise(False)
)

#show result
df_construct.printSchema()
Sleep_Disorder_counts = df_construct2.groupBy('BMI Status').count().orderBy('count', ascending=False)
Sleep_Disorder_counts.show()
```

```
root
 |-- Person ID: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Occupation: string (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Quality of Sleep: integer (nullable = true)
 |-- Physical Activity Level: integer (nullable = true)
 |-- Stress Level: integer (nullable = true)
 |-- BMI Category: string (nullable = true)
 |-- Blood Pressure: string (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Health: string (nullable = true)
 |-- BMI Status: boolean (nullable = false)
 |-- Sleep Disorder: boolean (nullable = false)
```

```
+----------+-----+
|BMI Status|count|
+----------+-----+
|      true|  217|
|     false|  158|
+----------+-----+
```

Figure 34: BMI Status column

Overall, deriving new attributes for classifying 'Sleep Health' and 'BMI Classification' using data type Boolean aligns well with the study objectives. It simplifies analysis, improves interpretability, and supports predictive modelling and recommendations.

3.4. Integrate various data sources

Due to limitations in the dataset, merging or appending another dataset to the existing one is not possible. Therefore, I manually extracted attributes age, occupation, and personal ID to a CSV file named "Sleep Health and Demographic" from the original dataset. To align with this simulation, age and occupation were excluded from the current data frame by dropping the columns, as shown in Figure 35. Age and Occupation have now been eliminated from the data frame. This is done to simulate a scenario in which one dataset focuses on information on an individual's lifestyle, sleep quality, and sleep health, while another dataset contains demographic information about the same individual.

In the context of two datasets available, 'Sleep_health_and_lifestyle_dataset(with_dirt). csv' and 'Sleep Health and Demographic.csv', each focusing on distinct aspects of the

same individual. Merging is the best technique to combine the two datasets. Merging is a method to combine multiple datasets into a single output, which is especially valuable when dealing with diverse data sources. By merging two datasets, we the increase available for data mining analysis.

To merge two datasets, first to import the new dataset as ‘df_demographic’, indicated by Figure 36. Next, merge the ‘df_demographic’ with data frame ‘df_1’ which initially sourced from 'Sleep_health_and_lifestyle_dataset(with_dirt).csv'. the merging used As ‘Person ID’ as the key to match information between the two datasets. This is due to the uniqueness of each individual's ID, allowing for seamless connection of information. For merging datasets, there are various methods available, but for this project, the inner join method has been selected. The inner join method ensures that only individuals present in both datasets are included in the new dataset, avoiding the introduction of new individuals and potential data inconsistencies. Figure 37 shows the data frame information after merging. Two new columns highlighted in a red box have been merged into the data frame, and each of the columns has 375 entries as shown in Figure 38. This means the merging was successful without introducing any additional entries.

```
#create a new DataFrame 'df_select' by dropping the 'Heart Rate' column
df_1 = df_construct2.drop('Age', 'Occupation')

#show schema
df_1.printSchema()
df_1.show(n=1)
```

```
root
 |-- Person ID: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Quality of Sleep: integer (nullable = true)
 |-- Physical Activity Level: integer (nullable = true)
 |-- Stress Level: integer (nullable = true)
 |-- BMI Category: string (nullable = true)
 |-- Blood Pressure: string (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Health: string (nullable = true)
 |-- Sleep Disorder: boolean (nullable = false)
 |-- BMI Status: boolean (nullable = false)

+---------+------+--------------+----------------+-----------------------+------------+------------+--------------+--
---------+------------+--------------+----------+
|Person ID|Gender|Sleep Duration|Quality of Sleep|Physical Activity Level|Stress Level|BMI Category|Blood Pressure|Da
ily Steps|Sleep Health|Sleep Disorder|BMI Status|
+---------+------+--------------+----------------+-----------------------+------------+------------+--------------+--
---------+------------+--------------+----------+
|        1|  Male|           6.1|               6|                     42|           6|  Overweight|        126/83|
4200|        None|         false|     false|
+---------+------+--------------+----------------+-----------------------+------------+------------+--------------+--
---------+------------+--------------+----------+
only showing top 1 row
```

Figure 35: drop Age and Occupation from data frame

```
#import original dataset
df_demographic = spark.read.csv('Datasets/Sleep Health and Demographic.csv', inferSchema=True, header=True)

#show schema
df_demographic.printSchema()
df_demographic.show(n=1)
```

```
root
 |-- Person ID: integer (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Occupation: string (nullable = true)

+---------+---+----------------+
|Person ID|Age|      Occupation|
+---------+---+----------------+
|        1| 27|Software Engineer|
+---------+---+----------------+
only showing top 1 row
```

Figure 36: Dataset of demographic

```
#merge two datasets
df_merged = df_1.join(df_demographic, on="Person ID", how="inner")
df_merged.printSchema()
```

```
root
 |-- Person ID: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Quality of Sleep: integer (nullable = true)
 |-- Physical Activity Level: integer (nullable = true)
 |-- Stress Level: integer (nullable = true)
 |-- BMI Category: string (nullable = true)
 |-- Blood Pressure: string (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Health: string (nullable = true)
 |-- Sleep Disorder: boolean (nullable = false)
 |-- BMI Status: boolean (nullable = false)
 |-- Age: integer (nullable = true)
 |-- Occupation: string (nullable = true)
```

Figure 37: Merging two datasets

```
df_merged.describe().show()
```

```
+-------+-----------------+------+-----------------+------------------+------------------------+-----------------+-
-----------+--------------+-----------------+------------+-----------------+----------+
|summary|        Person ID|Gender|   Sleep Duration| Quality of Sleep|Physical Activity Level|     Stress Level|B
MI Category|Blood Pressure|      Daily Steps|Sleep Health|              Age|Occupation|
+-------+-----------------+------+-----------------+------------+-----------------+----------+-
-----------+--------------+-----------------+------------+-----------------+----------+
|  count|              375|   375|              375|             375|                     375|              375|
375|           375|              375|         375|              375|       375|
|   mean|            188.0|  null| 7.134933311462403| 7.314666666666667|      59.22666666666667| 5.381333333333333|
null|          null| 6816.842666666666|        null|            42.136|      null|
| stddev|108.39741694339399|  null|0.7965036810241664|1.1958812262546714|      20.83072425730269|1.7735951986568852|
null|          null|1615.7512430625304|        null|8.712285259247578|      null|
|    min|                1|Female|              5.8|                 4|                      30|                3|
Normal|        115/75|             3000|    Insomnia|               24|Accountant|
|    max|              375|  Male|              8.5|                 9|                      90|                8|
Overweight|       142/92|            10000| Sleep Apnea|               59|   Teacher|
+-------+-----------------+------+-----------------+------------+-----------------+----------+-
-----------+--------------+-----------------+------------+-----------------+----------+
```

Figure 38: Describe of merged data

3.5. Format the data as required

As the final step before building a model, formatting is crucial to ensure the dataset's cleanliness and alignment with the model's requirements. Once data selection and construction are complete, it's important to remove duplicate fields from the dataset. In this case, we need to drop 'BMI Category' and 'Sleep Pattern' since two new attributes

have been created from them to provide more insights for data analysis. The code snippet in Figure 39 identifies the two columns to be dropped from the merged data frame created in the previous step.

The models I plan to use are Classification and Clustering. Classification algorithms are used in supervised learning tasks, classification is used for prediction and usually used in medical diagnosis (Rajendiran, 2015). Clustering helps to identify groups of records with similar characteristics (Patel , Modi , & Sarvakar, 2014).

For classification and clustering algorithms in data mining, the data types commonly used can vary, but they should ideally be numeric or convertible to numeric data types. Examples include nominal, ordinal, interval, and ratio. However, in the dataset, the Blood Pressure data type is a string, and it appears as '126/83', where the first number represents the systolic pressure (126 mmHg) and the second number represents the diastolic pressure (83 mmHg). To make the Blood Pressure data more meaningful and interpretable, I propose format Blood Pressure from String to Integer by splitting it into two separate columns: 'Systolic' and 'Diastolic' blood pressure values.

The process of reformatting and splitting Blood Pressure is shown in Figure 40. The split function is used to separate the 'Blood Pressure' column into two parts based on the '/' delimiter, and the resulting values are casted into integers. Then, two new columns are created using the 'withColumn' method. Since we have already transformed the Blood Pressure data into 'Systolic' and 'Diastolic' values, the Blood Pressure column is removed from the dataset as shown in Figure 41.

During the data preparation step, there may be issues that can affect the quality of the models or the results in the data mining process. We can always perform iterations to ensure the model is robust.

```
#drpop BMI Category and Sleep Health
df_formated = df_merged.drop('BMI Category', 'Sleep Health')
df_formated.printSchema()
```

```
root
 |-- Person ID: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Quality of Sleep: integer (nullable = true)
 |-- Physical Activity Level: integer (nullable = true)
 |-- Stress Level: integer (nullable = true)
 |-- Blood Pressure: string (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Disorder: boolean (nullable = false)
 |-- BMI Status: boolean (nullable = false)
 |-- Age: integer (nullable = true)
 |-- Occupation: string (nullable = true)
```

Figure 39: Reformatting to clean dataset

```
from pyspark.sql.functions import split

# Split the 'Blood Pressure' column into 'Systolic' and 'Diastolic' columns
df_formated2 = df_formated.withColumn('Systolic', split(df['Blood Pressure'], '/')[0].cast('integer'))
df_formated3 = df_formated2.withColumn('Diastolic', split(df['Blood Pressure'], '/')[1].cast('integer'))

# Show the DataFrame with the new columns
df_formated3 .printSchema()
df_formated3.show(n=1)
```

```
root
 |-- Person ID: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Quality of Sleep: integer (nullable = true)
 |-- Physical Activity Level: integer (nullable = true)
 |-- Stress Level: integer (nullable = true)
 |-- Blood Pressure: string (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Disorder: boolean (nullable = false)
 |-- BMI Status: boolean (nullable = false)
 |-- Age: integer (nullable = true)
 |-- Occupation: string (nullable = true)
 |-- Systolic: integer (nullable = true)
 |-- Diastolic: integer (nullable = true)

+---------+------+--------------+---------------+----------------------+------------+--------------+-----------+---
----------+----------+---+----------------+--------+---------+
|Person ID|Gender|Sleep Duration|Quality of Sleep|Physical Activity Level|Stress Level|Blood Pressure|Daily Steps|Sle
ep Disorder|BMI Status|Age|       Occupation|Systolic|Diastolic|
+---------+------+--------------+---------------+----------------------+------------+--------------+-----------+---
----------+----------+---+----------------+--------+---------+
|        1|  Male|           6.1|              6|                    42|           6|        126/83|       4200|
false|     false| 27|Software Engineer|     126|       83|
+---------+------+--------------+---------------+----------------------+------------+--------------+-----------+---
----------+----------+---+----------------+--------+---------+
only showing top 1 row
```

Figure 40: Reformatting and splitting of Blood Pressure

```
df_formated4 = df_formated3.drop('Blood Pressure')
df_formated4.printSchema()
```

```
root
 |-- Person ID: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Quality of Sleep: integer (nullable = true)
 |-- Physical Activity Level: integer (nullable = true)
 |-- Stress Level: integer (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Disorder: boolean (nullable = false)
 |-- BMI Status: boolean (nullable = false)
 |-- Age: integer (nullable = true)
 |-- Occupation: string (nullable = true)
 |-- Systolic: integer (nullable = true)
 |-- Diastolic: integer (nullable = true)
```

Figure 41: Remove Blood Pressure

4. Data transformation

4.1. Reduce the data

Data preparation involves removing noise from the dataset, while data transformation is the process of converting or modifying raw data into a suitable format or structure for data analysis and modelling. The first step in data transformation is feature selection, which helps identify key fields in the dataset for predicting sleep health. In this case, I use the Random Forest method for feature selection. According to Leo Breiman who trademarked random forest algorithm (Breiman, 2001), random forest is an ensemble learning method that constructs multiple decision trees and combines their outputs to produce a final prediction. One of the key advantages of random forest is its ability to

measure feature importance during the tree-building process. It keeps track of the contribution of each feature towards making accurate predictions. This also helps in reducing overfitting, as the method only selects one feature in the data at a time, avoiding noise or irrelevant features present in the data.

However, random forest requires numerical data for its decision-making process. The first step for feature selection is to perform encoding on two String type data Gender and Occupatio. As shown in Figure 42, ‘Gender’ is transformed from String to numerical data by one-hot encoding. and ‘Occupation’ is transformed by label encoding, it assigns a unique integer label to each category of a feature. the Figure 42, also shows the output after encoding. it displays the new columns ‘GenderIndex’ and ‘OccupationIndex’, they are both Double which is a numerical datatype. Figure 43 shows the mapping of the For gender, "Female" is assigned to "1" and "Male" is assigned to "0". For "Occupation", 11 occupations are assigned labels from 0 to 10. Figure 44 shows the coding to remove non numerical variables and extra variables generated during encoding, the dataset is now ready for feature selection.

BMI Status and ‘Sleep Disorder are both Boolean type, and they also need to be encoded to numerical numbers. As shown in Figure 45, 0 for false and 1 for true.

```
# Define the stages of the transformation pipeline
# 1.transform 'Gender' using one-hot encoding
gender_indexer = StringIndexer(inputCol="Gender", outputCol="GenderIndex")
gender_encoder = OneHotEncoder(inputCol="GenderIndex", outputCol="GenderVec")

# 2.transform 'Occupation' using label encoding
occupation_indexer = StringIndexer(inputCol="Occupation", outputCol="OccupationIndex")

# Assemble the transformation stages into a pipeline
pipeline = Pipeline(stages=[gender_indexer, gender_encoder, occupation_indexer])

# Fit and transform the data using the pipeline
df_trans = pipeline.fit(df_formated4).transform(df_formated4)

# Show the transformed DataFrame
df_trans.show(n=1)
df_trans.printSchema()
```

```
+---------+------+--------------+----------------+------------------------+------------+-----------+--------------+---
-------+---+-----------------+--------+---------+-----------+-------------+---------------+
|Person ID|Gender|Sleep Duration|Quality of Sleep|Physical Activity Level|Stress Level|Daily Steps|Sleep Disorder|BMI
Status|Age|       Occupation|Systolic|Diastolic|GenderIndex|    GenderVec|OccupationIndex|
+---------+------+--------------+----------------+------------------------+------------+-----------+--------------+---
-------+---+-----------------+--------+---------+-----------+-------------+---------------+
|        1|  Male|           6.1|               6|                      42|           6|       4200|         false|
false| 27|Software Engineer|     126|       83|        0.0|(1,[0],[1.0])|            8.0|
+---------+------+--------------+----------------+------------------------+------------+-----------+--------------+---
-------+---+-----------------+--------+---------+-----------+-------------+---------------+
only showing top 1 row

root
 |-- Person ID: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Quality of Sleep: integer (nullable = true)
 |-- Physical Activity Level: integer (nullable = true)
 |-- Stress Level: integer (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Disorder: boolean (nullable = false)
 |-- BMI Status: boolean (nullable = false)
 |-- Age: integer (nullable = true)
 |-- Occupation: string (nullable = true)
 |-- Systolic: integer (nullable = true)
 |-- Diastolic: integer (nullable = true)
 |-- GenderIndex: double (nullable = false)
 |-- GenderVec: vector (nullable = true)
 |-- OccupationIndex: double (nullable = false)
```

Figure 42: Encoding for Occupation and Gender

```
# Extract the mapping of numerical values to original categories
gender_mapping = df_trans.select("Gender", "GenderIndex").distinct()
occupation_mapping = df_trans.select("Occupation", "OccupationIndex").distinct()

# Show the mappings
gender_mapping.show()
occupation_mapping.show()
```

```
+------+-----------+
|Gender|GenderIndex|
+------+-----------+
|  Male|        0.0|
|Female|        1.0|
+------+-----------+

+--------------------+---------------+
|          Occupation|OccupationIndex|
+--------------------+---------------+
|           Scientist|            7.0|
|             Teacher|            4.0|
|             Manager|           10.0|
|              Doctor|            1.0|
|   Software Engineer|            8.0|
|              Lawyer|            3.0|
|         Salesperson|            6.0|
|Sales Representative|            9.0|
|            Engineer|            2.0|
|          Accountant|            5.0|
|               Nurse|            0.0|
+--------------------+---------------+
```

Figure 43: Mapping for gender and occupation

```
#drop extra columns
df_trans2 = df_trans.drop('Gender', 'GenderVec', 'Occupation')
df_trans2.printSchema()
```

```
root
 |-- Person ID: integer (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Quality of Sleep: integer (nullable = true)
 |-- Physical Activity Level: integer (nullable = true)
 |-- Stress Level: integer (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Disorder: boolean (nullable = false)
 |-- BMI Status: boolean (nullable = false)
 |-- Age: integer (nullable = true)
 |-- Systolic: integer (nullable = true)
 |-- Diastolic: integer (nullable = true)
 |-- GenderIndex: double (nullable = false)
 |-- OccupationIndex: double (nullable = false)
```

Figure 44: remove extra columns in dataset

```
# Encode "Sleep Disorder" as 0 for False and 1 for True
df_trans3 = df_trans2.withColumn(
    "Sleep Disorder",
    when(df_trans2["Sleep Disorder"] == True, 1).otherwise(0)
)

# Encode "BMI Status" as 0 for False and 1 for True
df_trans4 = df_trans3.withColumn(
    "BMI Status",
    when(df_trans3["BMI Status"] == True, 1).otherwise(0)
)
```

Figure 45: Encode for BMI and sleep disorder

As shown in Figure 44, after encoding, import ‘RandomForestClassifier’ and ‘VectorAssembler’class to be able to use the random forest algorithm. The data is assembled to single vector column, and then label Sleep Disorder and train the Random Forest Classifier. After that list the feature by feature importance from the trained model and print out the result.

The Figure 45 shows the feature importance of the DataFrame, which represents the relative importance of each feature in making predictions within a machine learning model. The numbers are scaled between 0 and 1, with 1 indicating the highest importance. According to the model, BMI status (0.267) has the highest importance, suggesting a strong relationship with the target variable 'Sleep Disorder'. The second importance is OccupationIndex with 0.24 importance. This is followed by Diastolic and Systolic with 0.18 and 0.12 respectively. Age (0.06), Person ID (0.05), and Daily Steps (0.03) and Sleep Duration (0.02) have a moderate level of importance, indicating some influence on the model's predictions. However, 'Person ID' is primarily used for identification purposes and is not meaningful for prediction. On the other hand, Physical Activity Level (0.016), 'Quality of Sleep' (0.016), 'Stress Level' (0.003), and 'Gender' (0.001) have lower importance compared to other features.

After recognising the significance of the features, I am considering the application of Occam's Razor, which emphasises simplicity. This is particularly important because collecting every detail about patients' lifestyle, demographic, and sleep patterns in the real world is impractical. In this scenario, data mining will prioritise the following features: BMI Status, OccupationIndex, Diastolic, Systolic, Age, Sleep Duration and Daily Steps as they have relatively higher importance values. Figure 46 depicts the code and DataFrame after feature selection, with the DataFrame now consisting of only seven features.

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import RandomForestClassifier

# 1. Assemble Features
feature_columns = ["Sleep Duration", "Quality of Sleep", "Physical Activity Level", "Stress Level",
                   "Daily Steps", "Age", "Systolic", "Diastolic", "GenderIndex", "OccupationIndex",
                   "BMI Status", "Person ID"]

vec_assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")
df_assembled = vec_assembler.transform(df_trans4)

# 2. Train a Random Forest Classifier
rf = RandomForestClassifier(featuresCol="features", labelCol="Sleep Disorder")
rf_model = rf.fit(df_assembled)

# 3. Extract Feature Importance
feature_importances = rf_model.featureImportances

# Create a list of (feature, importance) pairs
feature_importance_pairs = list(zip(feature_columns, feature_importances))

# Sort the pairs by importance in descending order
sorted_feature_importance_pairs = sorted(feature_importance_pairs, key=lambda x: x[1], reverse=True)

# Print the sorted features and their importance
for feature, importance in sorted_feature_importance_pairs:
    print(f"Feature: {feature}, Importance: {importance}")
```

```
Feature: BMI Status, Importance: 0.26681092816577734
Feature: OccupationIndex, Importance: 0.23684497996265125
Feature: Diastolic, Importance: 0.17891961497655615
Feature: Systolic, Importance: 0.12086987658454755
Feature: Age, Importance: 0.06447356340332848
Feature: Person ID, Importance: 0.04813593947602131
Feature: Daily Steps, Importance: 0.025460071021316134
Feature: Sleep Duration, Importance: 0.022015873298836752
Feature: Physical Activity Level, Importance: 0.016173917759716137
Feature: Quality of Sleep, Importance: 0.015988947913145146
Feature: Stress Level, Importance: 0.0032342166750770145
Feature: GenderIndex, Importance: 0.00107207076302669
```

Figure 46: Feature importance of DataFrame

```
#select features
#select the desired columns
selected_columns = ["BMI Status", "OccupationIndex", "Diastolic", "Systolic", "Age",
                    "Sleep Duration", "Daily Steps", "Sleep Disorder"]

#create the new DataFrame df_selected
df_selected = df_trans4.select(selected_columns)

#verify the schema of df_selected
df_selected.printSchema()
```

```
root
 |-- BMI Status: integer (nullable = false)
 |-- OccupationIndex: double (nullable = false)
 |-- Diastolic: integer (nullable = true)
 |-- Systolic: integer (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Disorder: integer (nullable = false)
```

Figure 47: Data reduction after feature selection

4.2. Project the data

It is important to have a balanced data distribution of the target field in a prediction study. Balancing the data ensure unbiased and accurate model outcomes, as the model may tend to favour the majority proportion over the minority.

The data value count in Figure 48 indicates that there are more individuals without a sleep disorder (220) than individuals with a sleep disorder (155) in the total dataset. To balance the proportion, there are two approaches: reducing (undersampling) and

boosting (oversampling). I have chosen to boost the mode because reducing it would result in data loss. While reducing can be effective for large datasets, the dataset only consists of 375 records, making boosting more suitable. Boosting involves synthetically creating data by referencing the existing distribution of other variables.

As shown in Figure 48, the "Sleep Disorder" variable is separated into majority (False) and minority classes (True). **Error! Reference source not found.** demonstrates the process of boosting the minority group through oversampling to match the majority class. Firstly, two DataFrames are created for each class. Then, the value count of each class is calculated, along with the oversampling ratio obtained by dividing the majority (0) count by the minority (1) count. Next, the minority class is oversampled using the calculated ratio. Finally, the two classes are combined into a new DataFrame. After the boosting process, the value count for the minority class increases from 155 to 217, which is very close to the count of the majority class. Figure 50 also includes a histogram plot of the 'Sleep Disorder' variable before and after boosting. Prior to boosting, the count of True was lower than the count of False. However, after boosting, the count of True becomes equal to the count of False.

```
#count Sleep Disorder
#group by the "Sleep Disorder" column and count occurrences of each parameter
sleep_disorder_counts = df_selected.groupBy("Sleep Disorder").count()

# Show the parameters and their counts
sleep_disorder_counts.show()
```

```
+--------------+-----+
|Sleep Disorder|count|
+--------------+-----+
|             1|  155|
|             0|  220|
+--------------+-----+
```

Figure 48: Value count of sleep disorder before balance

```
# Separate the DataFrame into two DataFrames for each class
df_class_0 = df_selected.filter(col("Sleep Disorder") == 0)
df_class_1 = df_selected.filter(col("Sleep Disorder") == 1)

# Calculate the oversampling ratio
class_0_count = df_class_0.count()
class_1_count = df_class_1.count()
oversampling_ratio = class_0_count / class_1_count

# Oversample the minority class (1)
df_class_1_oversampled = df_class_1.sample(withReplacement=True, fraction=oversampling_ratio, seed=42)

# Concatenate the oversampled minority class with the majority class
df_balanced = df_class_0.union(df_class_1_oversampled)

# Verify the count of each class in the balanced DataFrame
df_balanced.groupBy("Sleep Disorder").count().show()
```

```
+--------------+-----+
|Sleep Disorder|count|
+--------------+-----+
|             0|  220|
|             1|  217|
+--------------+-----+
```

Figure 49: Data boosting for 'Sleep Disorder'

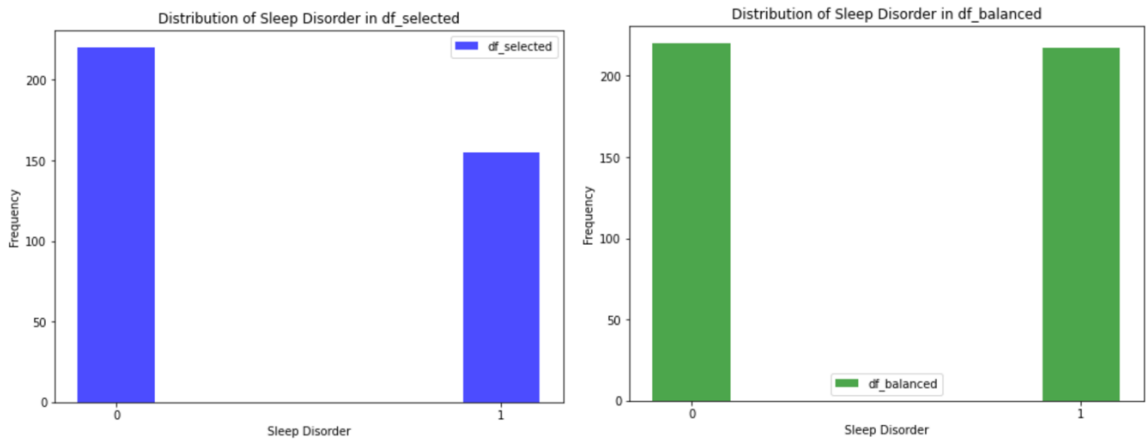

Figure 50: Plot bar of Sleep Disorder before (left) and after (right) boost balancing

5. Data-mining method(s) selection

5.1. Match and discuss the objectives of data mining (1.1) to data mining methods

Selecting the appropriate data mining method is an important step that aligns the chosen techniques with the data types and study objectives. This decision not only guides the modelling process but also enhances transparency and reproducibility through well-documented decision-making.

Data Types Availability:

As shown in Figure 51, the dataset contains integer, double, and float data types. Integer data includes BMI, Diastolic, Systolic, Age, Daily Steps, and Sleep Disorder, all of which are whole numbers without any decimal part. Float data includes Sleep Duration, which are floating-point numbers that can have decimal points. Double data consists of OccupationIndex, which can also have decimal places like float but provides more precision than a float.

```
df_balanced.printSchema()
```

```
root
 |-- BMI Status: integer (nullable = false)
 |-- OccupationIndex: double (nullable = false)
 |-- Diastolic: integer (nullable = true)
 |-- Systolic: integer (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Sleep Duration: float (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Sleep Disorder: integer (nullable = false)
```

Figure 51:Data info of balanced data

Data Mining Goals:

One of the goals of data mining has been addressed through feature selection (Step 4). This step identified crucial factors for robust analysis, improving data quality and paving the way for model development. The remaining goals involve predictive modelling to forecast sleep disorders and the creation of recommendation systems. To ensure accessibility and facilitate further research, it's crucial to present the study's outcomes in a manner that's easily understandable, benefiting both researchers and stakeholders.

Specific modelling requirements:

The study's objectives align with two principal methods: supervised learning (for prediction) and unsupervised learning (for personalised recommendation). Classification involves mapping inputs to outputs using pre-existing input-output pairs and is commonly applied in predictive modelling (Sarker, 2021). In this study, classification can be used to forecast sleep disorders based on historical patterns. The primary data types in classification are categorical, numeric, and binary data. Regression, another type of supervised learning, relies on a statistical model to understand the relationship between the dependent variable (target) and independent variables (features) (Sarker, 2021). Within the study, regression can predict sleep disorders based on feature analysis. Regression models require quantitative dependent and independent variables (What is linear regression, n.d.). Clustering, on the other hand, is an unsupervised learning method that identifies and groups related data points together based on their features (Sarker, 2021). In this study, a clustering model can help discover groups or clusters of individuals with similar characteristics to provide tailored recommendations to enhance sleep health. Clustering models can work with both numerical and categorical data, depending on the algorithms used (Clustering, 2021). For both supervised and unsupervised learning, the dataset should contain suitable data types to ensure the effectiveness of the chosen machine learning techniques.

5.2. Select the appropriate data-mining method(s) based on discussion

Based on the modelling requirements, data attributes, data mining goals, and success criteria, two methods stand out as relevant to this study: classification and clustering. classification models aim to identify individuals at risk of sleep disorders, while clustering methodologies group individuals for tailored recommendations. These selections are directly aligned with the data mining goals and success criteria, resulting in meaningful insights and actionable outcomes.

6. Data-mining algorithm(s) selection

6.1. Conduct exploratory analysis and discuss

The data mining objectivs involves relationship exploration, classification and clusteering for sleep health analysis.

For prediction tasks in classification algorithms, the focus will be on evaluating algorithms that can construct accurate models for predicting sleep-related outcomes. This includes algorithms such as Single Decision Tree which builds a tree-like structure to make decisions based on input features. Random Forests, are able to handle complex relationships by consisting of multiple decision trees. Gradient Boosted Trees (GBT) is a powerful and popular classification and regression method, it combines the predictions of multiple decision trees to create a more accurate and robust model (Sarker, 2021).

When comparing Single Decision Tree, Random Forests, and GBT, Single Decision Tree provides interpretable results that are easily understood through the coefficients of the features. However, it may overfit high-dimensional datasets and works well on small to moderate-sized datasets (Sarker, 2021). Random Forests deliver high predictive accuracy and minimise the overfitting problem (Sarker, 2021), but the feature importance scores they provide are less interpretable compared to Single Decision Tree. GBT are suitable for complex datasets and Resistant to overfitting due to boosting process (Sarker, 2021), However, GBT is sensitive to hyperparameters such as learning rate and tree depth, and is less interpretable compared to simpler models like Single Decision Tree.

When it comes to clustering algorithms, our focus will be on exploring algorithms that can effectively group individuals based on their sleep metrics and habits. One

algorithm is K-means clustering, which can be used to cluster individuals based on their sleep patterns, lifestyle habits, and demographic factors. This can help identify groups with similar sleep health characteristics (Sarker, 2021). Another algorithm is Hierarchical Clustering, which provides a hierarchical view of clusters. Comparing K-means and Hierarchical Clustering, K-means is computationally efficient with large datasets and easy to implement and apply. However, the number of clusters needs to be specified beforehand, which may not always be known. On the other hand, Hierarchical Clustering does not require specifying the number of clusters in advance, and the hierarchical tree structure provides more informative results compared to the flat clusters of K-means. However, Hierarchical Clustering is not suitable for computationally intensive and large datasets (Sarker, 2021).

6.2. Select data-mining algorithms based on discussion

Based on the exploration of data mining algorithms in section 6.1, I have selected Single Decision Tree for classification and K-means for clustering. This decision is influenced by availability of PySpark MLib library and the size of my dataset, which consists of only 375 entries, making it small to moderately sized.

Single Decision Tree is a suitable choice as it is simple and provides a good understanding of the data. It was discussed in section 6.1 that Single Decision Tree has higher interpretability compared to other algorithms. In the context of predicting patients' sleep disorders, interpretability is crucial for clinical decision-making. It is important to have clear and understandable factors that influence the predictions.

I selected K-means over Hierarchical Clustering for this study because the PySpark library does not support Hierarchical Clustering. K-means is a fast, robust, and simple algorithm that divides the data into non-overlapping clusters. The user needs to specify the number of clusters, which may require some experimentation or experience. K-means is widely used in the real world, such as for anomaly detection, to group similar data points together. It is highly useful for decision making.

6.3. Build/Select appropriate model(s) and choose relevant parameter(s)

Single Decision Tree model and parameters:

Following algorithm exploration in section 6.1 and algorithm selection in section 6.2, the choice for constructing a model for data mining and prediction on the prepared dataset is the Single Decision Tree. Single Decision Tree 's exceptional interpretability ensures that we gain clear and comprehensible insights into the factors shaping our predictions. The model is designed with the target variable 'Y' denoting Sleep Disorder and independent variables 'X' including Age, BMI Status, Sleep Duration, Occupation, and Daily Steps, as shown in **Error! Reference source not found.**.

Selecting suitable parameters is crucial for the performance of the Single Decision Tree model. In Single Decision Tree model, few parameters are considered vital important such as maxDepth and maxBins (Mithrakumar, 2019).

The maxDepth parameter determines the maximum depth of the decision tree, controlling how deep the tree can grow. Smaller values limit the tree's depth to prevent overfitting, but this may lead to underfitting. Conversely, larger values allow for a deeper tree, but there is a risk of overfitting. The maxBins parameter determines the maximum number of bins used to discretize continuous features. Increasing this value can improve model accuracy, but it may also increase computation time and memory usage.

For my dataset, I have decided to use the default value of 5 for maxDepth and the default value of 32 for maxBins. Default values are a good starting point for my dataset, although they are not guaranteed to be the best for your specific dataset. To find the optimal value for the hyperparameter, it may need to use cross-validation. This can be achieved in future iterations.

K-means model and parameters:

After exploring algorithms in section 6.1 and selecting algorithm in section 6.2, the K-means algorithm is the chosen method for building a model in data mining to offer personalised recommendations for improving sleep health.

When building K-means models, the most important parameter is K, which represents the number of clusters. Since I had no prior experience in this domain, I decided to

use the Silhouette method to determine the optimal value for K. The silhouette score measures the similarity of each data point to its own cluster compared to other clusters. A score close to 1 indicates that the data points are well-clustered, while a score close to -1 suggests that the data points are closer to neighbouring clusters rather than their own, indicating an incorrect assignment of the k value (Bhardwaj, 2020).

In Figure 52, the code to find the best K in the dataset is shown. It loops through k values from 2 to 11 and determines that 10 is the best K with a score of 0.98. This means that when the K value is 10, the cluster is well structured and distinct from neighbouring clusters.

```
#find the best optimal k
from pyspark.ml.evaluation import ClusteringEvaluator

silhouette_scores = []
for k in range(2, 11):
    kmeans = KMeans(k=k, seed=42)
    model = kmeans.fit(df_assembled)
    predictions = model.transform(df_assembled)
    evaluator = ClusteringEvaluator()
    silhouette_score = evaluator.evaluate(predictions)
    silhouette_scores.append(silhouette_score)

best_k = silhouette_scores.index(max(silhouette_scores)) + 2  # +2 because we started from k=2
print("best K is: ", best_k, " score: ", max(silhouette_scores))
```

```
best K is:  10  score:  0.982340902677423
```

Figure 52: Finding max K value

7. Data Mining

7.1. Create and justify test designs

Before conducting data mining, it's essential to establish a testing design to evaluate the performance of the machine learning model and avoid overfitting. Evaluating performance requires splitting the data into two sets: one for training and the other for testing. The majority of the data is allocated to training, where models are constructed. The remaining data is reserved for testing, allowing us to assess the model's performance on unseen data and ensure that the model doesn't memorise the training data.

Common train-test ratios, such as 80:20 and 70:30, are frequently used in machine learning. When there is less training data, parameter estimates tend to exhibit higher variability. Conversely, a shortage of testing data can lead to greater variability in performance metrics (Tokuc, 2023).

With a relatively small dataset of 375 samples, an 80:20 ratio is chosen for this study. This allocation provides a decent amount of data for training while still reserving a portion for testing. As shown in Figure 53, the DataFrame is divided into training and testing sets using an 80:20 split. Figure 54 shows the details of this split. The total size of sample is 437, with the testing set comprising 76 data points (17% of the total size) and the training ser containing 361 data points (83% of the total size).

```
# Split the data into 80% training and 20% testing
(training_data, test_data) = df_assembled.randomSplit([0.8, 0.2], seed=42)
```

Figure 53:Coding for splitting dataset into 80:20

```
# Count the number of samples in the training set
training_count = training_data.count()
print(f"Number of samples in the training set: {training_count}")

# Count the number of samples in the testing set
testing_count = test_data.count()
print(f"Number of samples in the testing set: {testing_count}")
```

```
Number of samples in the training set: 361
Number of samples in the testing set: 76
```

Figure 54: Data size after splitting into training and testing

7.2. Conduct data mining – classify, regress, cluster, etc. (models must execute)

Single Decision Tree model:

After dividing the dataset into training and testing subsets, the single decision tree model is built following the code shown in Figure 55, Initially, seven essential features are defined in a column. Then, a vector assembler is created to assemble the features into a feature vector. After that, the dataset is split into training and testing with a ratio of 80:20, as discussed in section 7.1 on test design. Next, a decision tree classifier is created, with the Sleep Disorder column set as the label column, which represents the target for prediction. The model is then trained using the 'fit' function. Once the model is trained, the test set is used to perform predictions using the 'transform' function.

Figure 56 shows evaluation of the model, the area under the curve (AUC) is 0.9 and model accuracy is 92.11%.

```
#import library for decision tree classifier
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

```
# Define the feature columns (excluding the target column)
feature_columns = ['BMI Status', 'OccupationIndex', 'Diastolic', 'Systolic', 'Age', 'Sleep Duration', 'Daily Steps']

# Create a VectorAssembler to assemble the feature columns into a single feature vector
vector_assembler = VectorAssembler(inputCols=feature_columns, outputCol='features')
df_assembled = vector_assembler.transform(df_balanced)
```

```
# Split the data into 80% training and 20% testing
(training_data, test_data) = df_assembled.randomSplit([0.8, 0.2], seed=42)
```

```
# Create the Decision Tree classifier
dt = DecisionTreeClassifier(labelCol='Sleep Disorder', featuresCol='features', seed=42)

# Train the model
dt_model = dt.fit(training_data)
```

```
# Make predictions on the test data
predictions = dt_model.transform(test_data)
```

Figure 55: Build model for single decision tree model

```
# Evaluate the model using BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator(labelCol='Sleep Disorder', metricName='areaUnderROC')
auc = evaluator.evaluate(predictions)
print(f"AUC: {auc}")
```

```
AUC: 0.90406162464986
```

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Create a MulticlassClassificationEvaluator
evaluator_multiclass = MulticlassClassificationEvaluator(
    labelCol='Sleep Disorder',
    predictionCol='prediction',
    metricName='accuracy'  # You can choose other metrics like 'accuracy', 'weightedPrecision', 'weightedRecall', etc.
)
```

```
# Calculate the evaluation metric
accuracy_score = evaluator_multiclass.evaluate(predictions)
print('The Single Decision Tree Model has an accuracy of: {0:2.2f}%'.format(accuracy_score*100))
```

```
The Single Decision Tree Model has an accuracy of: 92.11%
```

Figure 56:evaluation of single decision tree model

K-means:

The model's cluster number was experimented, and the best value for K was found to be 10. As shown in Figure 57, , all features, including Sleep Disorder, were selected. A Vector Assembler was then created, similar to the Single Decision Tree model. Next, a K-means model was created with K set to 10. The model was built using the 'fit' function, and predictions were made using the 'transform' function. Evaluation of the model is shown in Figure 58 and Figure 59. These figures provide details about the 10 clusters and the number of samples in each cluster. Cluster 1 is the largest cluster, consisting of 106 samples, while clusters 7, 8, and 9 are the smallest, with only 4 samples each.

```
#Select the relevant columns for your K-means model
selected_columns_K = ["BMI Status", "OccupationIndex", "Diastolic", "Systolic", "Age",
                      "Sleep Duration", "Daily Steps", "Sleep Disorder"]

#Create a VectorAssembler to combine selected columns into a feature vector
assembler_K = VectorAssembler(inputCols=selected_columns_K, outputCol="features")
df_assembled_K = assembler_K.transform(df_balanced)
```

```
# Create a K-Means model
kmeans = KMeans(k=10, seed=42)

# Fit the K-Means model to the data
model = kmeans.fit(df_assembled)
```

```
# Make predictions using the model
predictions_K = model.transform(df_assembled)
```

```
# View the clustering results
predictions_K.select("features", "prediction").show()

# Show the cluster centers
print("Cluster Centers:")
centers = model.clusterCenters()
for center in centers:
    print(center)
```

Figure 57: Coding for K-means clustering

```
# View the clustering results
predictions_K.select("features", "prediction").show()

# Show the cluster centers
print("Cluster Centers:")
centers = model.clusterCenters()
for center in centers:
    print(center)
```

```
+--------------------+----------+
|            features|prediction|
+--------------------+----------+
|[0.0,8.0,83.0,126...|         7|
|[1.0,1.0,80.0,125...|         2|
|[1.0,1.0,80.0,125...|         2|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
|[1.0,1.0,80.0,120...|         1|
+--------------------+----------+
only showing top 20 rows

Cluster Centers:
[2.02020202e-02 4.89898990e+00 8.70202020e+01 1.31929293e+02
 4.45050505e+01 6.53636365e+00 6.00202020e+03]
[1.00000000e+00 2.16037736e+00 8.29245283e+01 1.25849057e+02
 3.64056604e+01 7.50283017e+00 8.00000000e+03]
[8.69565217e-02 8.69565217e-02 9.34782609e+01 1.38391304e+02
 4.90869565e+01 6.16086949e+00 1.00000000e+04]
[0.00000000e+00 2.75000000e+00 9.03125000e+01 1.39250000e+02
 4.08750000e+01 7.14375016e+00 3.38750000e+03]
[8.73239437e-01 1.67605634e+00 8.11830986e+01 1.26183099e+02
 4.33521127e+01 7.36056333e+00 4.99718310e+03]
[4.67532468e-01 2.10389610e+00 8.56883117e+01 1.28428571e+02
 4.78051948e+01 7.68701296e+00 6.98981818e+03]
[1.66666667e-01 5.83333333e+00 8.41666667e+01 1.27500000e+02
 3.66666667e+01 6.59999990e+00 5.53333333e+03]
[5.000e-01 1.500e+00 8.575e+01 1.300e+02 3.250e+01 6.375e+00 4.075e+03]
[1.0000000e+00 5.0000000e+00 7.7500000e+01 1.1700000e+02 3.9500000e+01
 7.5999999e+00 7.4000000e+03]
[   0.            7.           86.          131.           34.
    5.80000019 5200.        ]
```

Figure 58: Clustering results

```
#counts in different prediction group
cluster_counts = predictions_K.groupBy("prediction").count()
cluster_counts.show()
```

```
+----------+-----+
|prediction|count|
+----------+-----+
|         1|  106|
|         6|    6|
|         5|   77|
|         4|   71|
|         8|    4|
|         7|    8|
|         2|   46|
|         0|   99|
|         3|   16|
|         9|    4|
+----------+-----+
```

Figure 59: Clusters counts

7.3. Search for patterns

This section is to look for trends and correlations between variables and clusters from models.

Patterns from Feature Importance:

Figure 60 displays a heat-map that represents feature correlation from a balanced DataFrame. The colour intensity indicates the strength of correlation, and only correlations greater than 0.5 are shown. In the figure, stronger correlations are depicted with more saturated colours, such as dark red and dark blue.

As shown in the figure, it is evident that Systolic-Diastolic, Systolic/Diastolic-Sleep Disorder and Systolic/Diastolic-Age display positive correlation (**Pattern 1**). Conversely, BMI Status-Sleep Disorder, Systolic/Diastolic-BMI Status and BMI Status-Age show strong negative correlations **(Pattern 2)**. It is worth highlighting that Systolic-Diastolic has a positive relationship with other features, while BMI Status has a negative relationship with other features. This suggests that there may be significant insights within the clustering categories that warrant further investigation in subsequent steps.

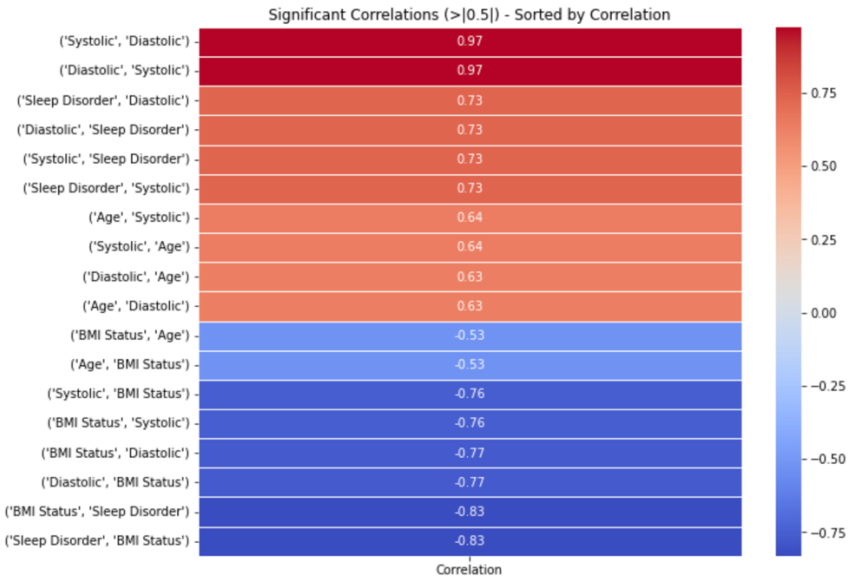

Figure 60: Heatmap of feature correlation from Logistic Regression

Patterns from Decision Tree: the Figure 61 shows the string presentation of the decision tree. 'depth=2' indicates that the tree has a maximum depth of 2 levels. 'numNodes=7' and 'numFeatures=7' indicate that the total number of nodes in the decision tree is 7, and the model uses 7 features for making decisions. 'numClasses=2' indicates that the model is a binary classification model, meaning it predicts one of two classes. The tree structure first checks whether the value of feature 0 is less than or equal to 0.5 (**Pattern3**). If it is, the tree splits into two nodes to check feature 3 and feature 2, respectively, to predict the result as 0 or 1. To gain more insights from this pattern, we need to mine additional information such as the names of feature 0, feature 2, and feature 3, as well as how the nodes work to make predictions.

```
print(tree_model_str)
```

```
DecisionTreeClassificationModel: uid=DecisionTreeClassifier_8b749269606d, depth=2, numNodes=7, numClasses=2, numFeatu
res=7
  If (feature 0 <= 0.5)
   If (feature 3 <= 128.5)
    Predict: 0.0
   Else (feature 3 > 128.5)
    Predict: 1.0
  Else (feature 0 > 0.5)
   If (feature 2 <= 85.5)
    Predict: 0.0
   Else (feature 2 > 85.5)
    Predict: 1.0
```

Figure 61: String presentation of Decision Tree

Patterns from Clustering model: as shown in Figure 62 the dataset is divided into 10 separate clusters using Principal Component Analysis (PCA). In this distribution, most

clusters exhibit similar values for Principal Component 1. Figure 63 provides a breakdown of data points in each cluster, with Cluster 1 being the largest and containing the most data points. However, there are smaller clusters like Cluster 8 and Cluster 9, which only have 4 data points each. It is worth investigating the similarities and differences between these clusters to understand what makes them distinct.

Figure 64 displays the average feature values for each cluster. When comparing Cluster 0 and Cluster 4 in Figure 65, they have similarities in Daily Steps, Sleep Duration, Age, Systolic, and Diastolic. However, they differ in Occupation and BMI. Cluster 0 has almost no value in BMI, while Cluster 4 has a value of around 1 in BMI. This suggests that Cluster 0 has a high occurrence of sleep disorders, while Cluster 4 has a lower occurrence. A similar pattern can also be observed when comparing Cluster 0 and Cluster 6 in Figure 66. These two clusters are similar in almost all features, except for Occupation and BMI (**Pattern4**). Further insights can be gained through visualization in the later section.

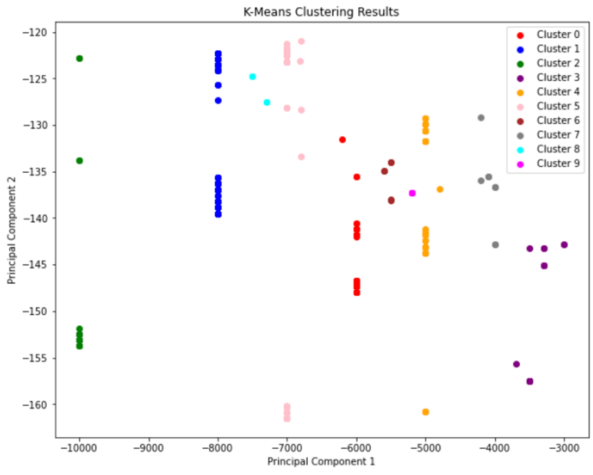

Figure 62:Distribution of clusters in dataset by Principal Component Analysis (PCA)

prediction	count
0	99
1	106
2	46
3	16
4	71
5	77
6	6
7	8
8	4
9	4

Figure 63: Value counts in each cluster

prediction	Sleep_Disorder	Avg_BMI	Avg_Occupation	Avg_Dia	Avg_Systolic	Avg_Age	Avg_Sleep	Avg_Steps	Cluster_Size
0	0.92	0.02	4.90	87.02	131.93	44.51	6.54	6,002.02	99.00
1	0.12	1.00	2.16	82.92	125.85	36.41	7.50	8,000.00	106.00
2	0.87	0.09	0.09	93.48	138.39	49.09	6.16	10,000.00	46.00
3	1.00	0.00	2.75	90.31	139.25	40.88	7.14	3,387.50	16.00
4	0.14	0.87	1.68	81.18	126.18	43.35	7.36	4,997.18	71.00
5	0.48	0.47	2.10	85.69	128.43	47.81	7.69	6,989.82	77.00
6	0.00	0.17	5.83	84.17	127.50	36.67	6.60	5,533.33	6.00
7	0.75	0.50	1.50	85.75	130.00	32.50	6.38	4,075.00	8.00
8	0.00	1.00	5.00	77.50	117.00	39.50	7.60	7,400.00	4.00
9	1.00	0.00	7.00	86.00	131.00	34.00	5.80	5,200.00	4.00

Figure 64: Feature average index in different clusters

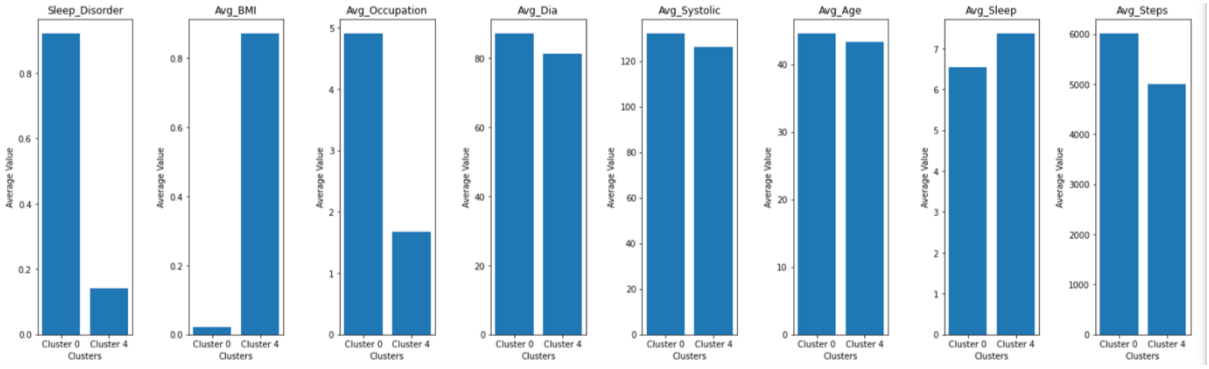

Figure 65: Compare cluster 0 and cluster 4

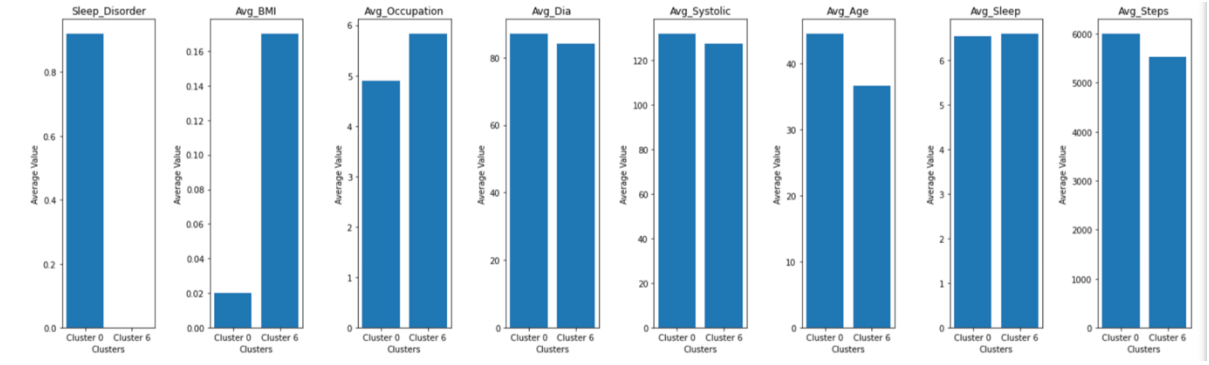

Figure 66: Compare cluster 0 and cluster 6

8. Interpretation

8.1. Study and discuss the mined patterns

Pattern 1: As discussed in section 7.2, there is a positive correlation between Systolic-Diastolic, Systolic/Diastolic-Sleep Disorder, and Systolic/Diastolic-Age. The correlation coefficient between Systolic-Diastolic is 0.97, and the correlation coefficient between Systolic/Diastolic-Sleep Disorder is 0.73. Both coefficients are close to 1, indicating a very strong correlation. This means that when Systolic increases, Diastolic and the risk of Sleep Disorder also increase. Systolic/Diastolic-Age has a correlation coefficient of 0.64, indicating a moderate correlation.

Conducting further visual analysis can provide more insights into the interrelationship between these three factors.

Pattern 2: According to section 7.2, there is a negative correlation between BMI Status and Sleep Disorder, Systolic/Diastolic and BMI Status, and BMI Status and Age. The correlation coefficient between BMI Status and Sleep Disorder is -0.83, and the correlation coefficient between Systolic/Diastolic and BMI Status is -0.77. Both coefficients indicate a very strong correlation close to -1. This suggests that as BMI decreases, the risk of Sleep Disorder and Systolic/Diastolic increases. The correlation coefficient between Systolic/Diastolic and Age is -0.53, indicating a moderate correlation. Further visual analysis can provide more insights into the interrelationship among these four factors.

Pattern 3: This pattern presents decision process in decision tree model. Figure 67 displays the feature number and its corresponding name. To integrate this list with Figure 61 it presents the decision tree process. The decision tree first checks the datapoint against its BMI status. If the BMI is false (<=0.5) and the systolic value is <= 128.5, it is predicted as not having a sleep disorder. If the systolic value is > 128.5, it is predicted as having a sleep disorder. On the other hand, if the BMI is positive and the Diastolic value is <= 85.5, the individual is predicted to be free from a sleep disorder. If the Diastolic value is > 85.5, the individual is predicted to have a sleep disorder. The tree model uses three features as nodes in its structure, and all 7 features work together to establish the structure, including the threshold values for diastolic and systolic numbers.

```
#Pattern 3
#Decision Tree presentation
for index, feature in enumerate(feature_columns):
    print(f'{index}: {feature}')
```

```
0: BMI Status
1: OccupationIndex
2: Diastolic
3: Systolic
4: Age
5: Sleep Duration
6: Daily Steps
```

Figure 67: numbers and features

Pattern 4: This pattern is identified through the clustering model. It involves comparing two clusters, both of which have similar characteristic in Daily Steps, Sleep Duration, Age, Systolic, and Diastolic. However, they exhibit differences in terms of Occupation, BMI. This suggests that the presence of a sleep disorder doesn't appear to be directly linked to factors like Daily Steps, Sleep Duration, Age, Systolic, and Diastolic, but it does seem to have a strong association with BMI Status and Occupation.

8.2. Visualise the data, results, models, and patterns

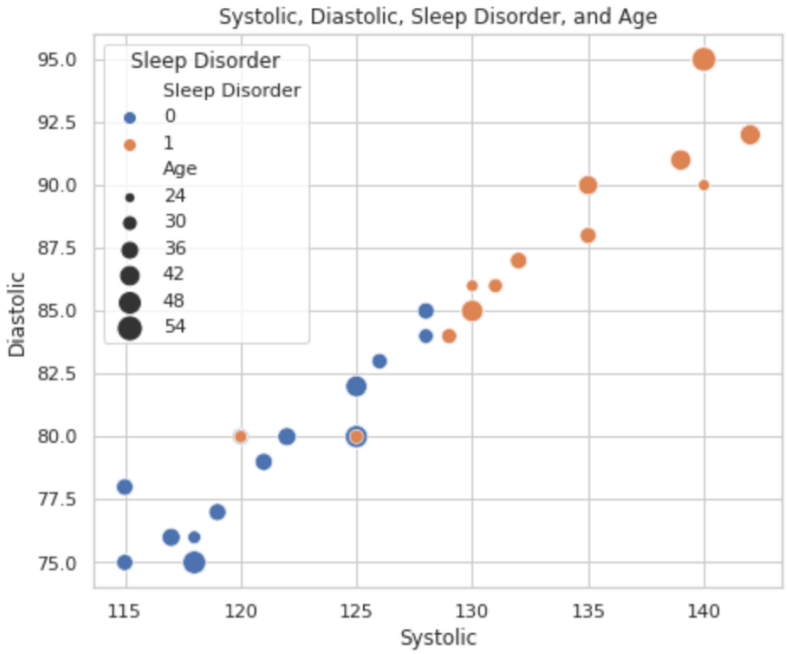

Figure 68: Scatterplot plot of Systolic, Diastolic, Sleep Disorder and Age

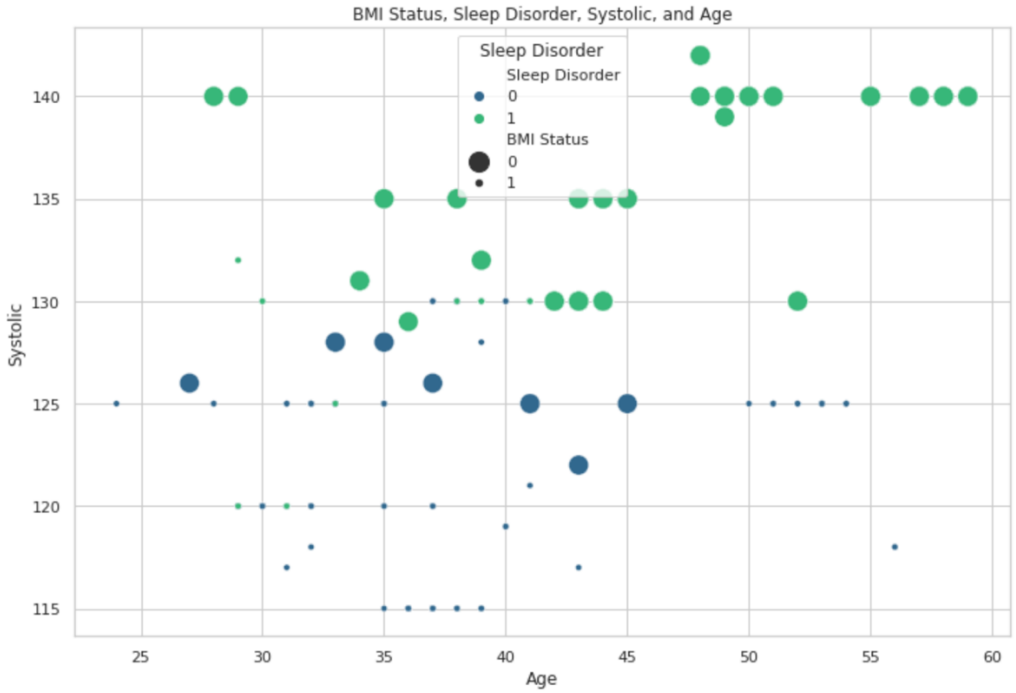

Figure 69: Scatterplot of BMI, Sleep Disorder, Diastolic and Age

BMI Status	Sleep Disorder	count
1	0	201
0	0	19
1	1	18
0	1	199

Figure 70: value counts of BMI Status and Sleep Disorder

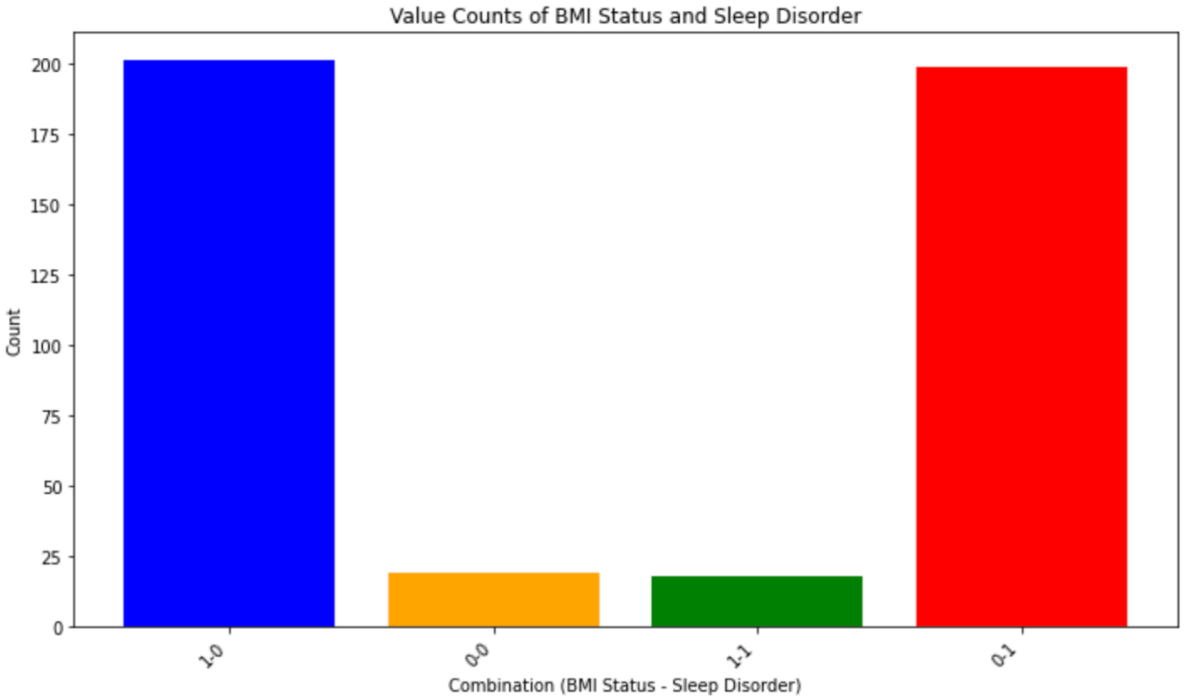

Figure 71: bar chart of value counts of BMI Status and Sleep Disorder

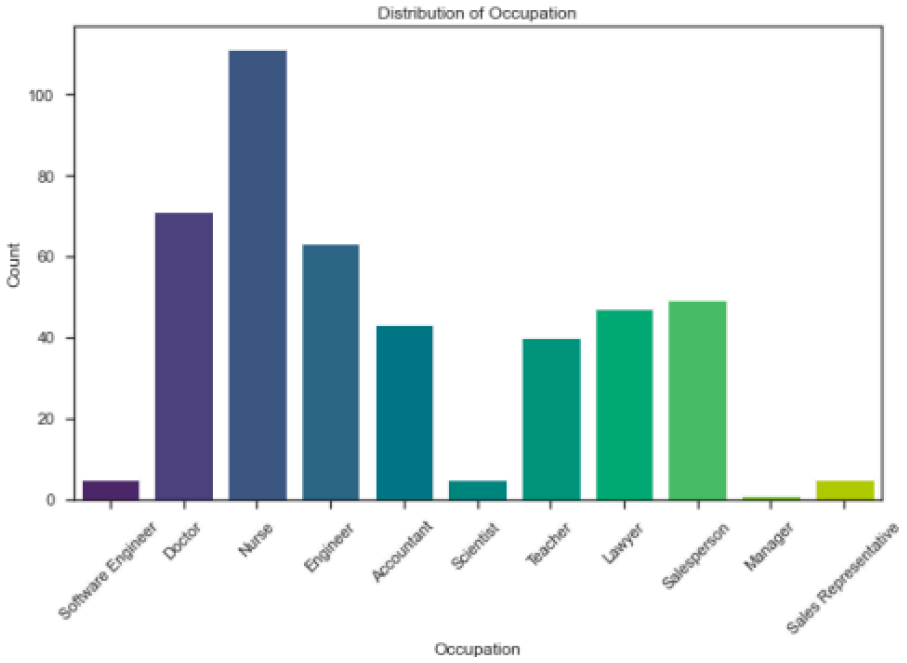

Figure 72: Distribution of Occupations l

8.3. Interpret the results, models, and patterns

Pattern 1: Figure 68 shows a scatterplot illustrating the relationship between Systolic, Diastolic, Sleep Disorder, and Age. The y-axis represents Diastolic, which increases along with the x-axis representing Systolic. This indicates a positive correlation between Diastolic and Systolic. It is evident from the plot that most of the blue points, representing the absence of sleep disorder, are located in the bottom-left side of the plot where systolic is less than approximately 128 and diastolic is less than 85. On the other hand, the yellow data points, representing the presence of sleep disorder, are mostly located in the top-right of the plot where systolic is greater than 128 and diastolic is greater than 85. The size of the data points represents the age of individuals, with larger points indicating greater age. However, the trends of positive correlation with age and systolic/diastolic are not very obvious from the figure. Therefore, this pattern indicates a positive correlation between systolic and diastolic. When systolic is smaller than 128 and diastolic is smaller than 85, it is more likely to have healthy sleep. Conversely, there is a higher risk of having a sleep disorder.

Pattern 2: Figure 69 displays a scatterplot illustrating the relationship between BMI status, sleep disorder systolic, and age. When systolic is greater than approximately 128, the data dots tend to be green. Conversely, when systolic is smaller than 128, the data dots are more likely to be blue. Additionally, when comparing dot sizes, most of the green dots are larger, while blue dots are mostly smaller. This indicates that individuals with abnormal BMI and systolic greater than 128 are more likely to have a sleep disorder. Conversely, individuals with normal BMI and systolic levels are more likely to have a healthy sleep pattern. Similar to Pattern 1, no obvious pattern can be observed in relation to age.

Pattern 3: Pattern 3 complements pattern 2 and pattern 1. The checking value in the tree node, as illustrated in the decision tree structure, is 128.5 for Systolic and 85.5 for diastolic. These values align with the figures for pattern 2 and pattern 3, emphasising their importance as indicators for predicting sleep disorders.

Pattern 4: Pattern 4 focuses on comparing two clusters that have similar features but differ in BMI and Occupation, leading to different predictions. To further investigate the relationship between BMI and Occupation with Sleep Disorder, I conducted additional research. In Figure 70 and Figure 71, the blue and red bars represent the

majority of individuals. The blue bar indicates a high number of individuals with normal BMI who have healthy sleep, while the red bar represents a large number of individuals with abnormal BMI who also have sleep disorders. This pattern paints a coherent picture that emphasises BMI Status as a critical influencing factor in the occurrence of sleep disorders.

Figure 72, provides an overview of the occupation distribution in the dataset. It's evident that the dataset's occupation distribution is not uniform, with Nurse, Doctors, and Engineering being the top three most common occupations. Nurse, for instance, the count of nurse present in sleep disorder is shown in Figure 73 where Sleep Disorder is equal to 1 and OccupationIndex it 0.0. after calculation, Nurse accounts for 29.6% of the total occupation counts, and within this group, 58.67% have a 'False' BMI Status. This means that Nurse makes up a substantial portion of the 'False' BMI Status category, reaching 46.36%. However, it's important to consider that the overrepresentation of Nurse in the sample may introduce bias into the data analysis. The assumptions about Occupation being more likely to have an Sleep Disorder should be approached cautiously, as it could be influenced by the imbalanced sample representation.

```
occupation_counts = predictions_K.groupBy("Sleep Disorder", "OccupationIndex").count()

# Show the occupation counts
```

```
+--------------+---------------+-----+
|Sleep Disorder|OccupationIndex|count|
+--------------+---------------+-----+
|             0|            2.0|   57|
|             0|            3.0|   42|
|             0|            0.0|    9|
|             0|            7.0|    2|
|             0|            8.0|    3|
|             0|            4.0|   10|
|             0|            5.0|   30|
|             0|           10.0|    1|
|             0|            1.0|   64|
|             0|            6.0|    2|
|             1|            6.0|   43|
|             1|            0.0|   85|
|             1|            7.0|    4|
|             1|            3.0|   12|
|             1|            9.0|    1|
|             1|            8.0|    1|
|             1|            5.0|    7|
|             1|            2.0|    5|
|             1|            1.0|   12|
|             1|            4.0|   47|
+--------------+---------------+-----+
```

Figure 73: Occupation counts in sleep disorder

8.4. Assess and evaluate results, models, and patterns

Single Decision Tree: The Figure 56 illustrates the performance of the Single Decision Tree model. The AUC is 0.90 and the model accuracy is 92.11%, indicating high performance. This result meets the study objective of achieving a minimum prediction accuracy rate of 85% for sleep disorder detection within the developed predictive models, as the accuracy exceeds 85%.

K-Means: The model's K number was experimented using the Silhouette method, as shown in Figure 52. The best value for K was found to be 10, with the highest score of 0.92. This score indicates that the clusters are well-structured and distinct from their neighbours, as it is very close to 1.

Overall assessment and evaluation:

The data mining process yielded two models: one employing supervised learning through Single Decision Tree, and the other utilizing unsupervised learning with K-Means. The Single Decision Tree model highlighted that BMI Status, Systolic and Diastolic plays a pivotal role in assessing the probability of experiencing a sleep disorder. Meanwhile, the Single Decision Tree model furnishes a highly accurate prediction model for identifying sleep disorders. After analysing the models and data, several patterns have emerged:

Pattern 1: Positive correlation between systolic and diastolic values for sleep disorder.
Pattern 2: Abnormal BMI and high systolic values indicate sleep disorder risk.
Pattern 3: Decision tree highlights the importance of systolic and diastolic values for predicting sleep disorders.
Pattern 4: Abnormal BMI is associated with sleep disorders, caution needed with occupation assumptions. These patterns provide valuable insights and can be utilised to improve understanding and management of sleep disorders.

Overall, the data mining process successfully achieved its objectives of exploring relationships, predictive modelling, and clustering for sleep health analysis. The

models and findings can be used to provide insights and recommendations for individuals with sleep health issues.

Further data mining work could involve improving accuracy of clustering model, exploring the relationships between sleep-related factors and other health conditions, such as mental health or chronic diseases. Additionally, investigating the impact of lifestyle factors, such as diet and exercise, on sleep health could provide further insights for individuals looking to improve their sleep habits.

8.5. Iterate prior steps (1 – 7) as required

Based on the assessment from the previous step, we will focus on improving model accuracy. In the feature selection part, we had previously selected 7 features BMI Status, OccupationIndex, Diastolic, Systolic, Age, Sleep Duration and Daily Steps. For prediction in decision tree model. However, I am going to use Kitchen sink approach to include all the features available this time to include Stress Level, Quality of Sleep Physical Activity Level and GenderIndex as shown in Figure 74.

After running the Single Decision Tree model, the AUC and accuracy increased to 0.978 and 97.37% respectively, as shown in Figure 75 and Figure 76. Both figures demonstrate a dramatic increase and illustrate the high performance of the model. The decision tree's node has also changed, with quality of sleep replacing diastolic. Similarly to the previous iteration, BMI status remains the first node to check in the decision tree.

In the second iteration, the inclusion of physical activity data in the dataset significantly improved the model's accuracy. This addition also provided additional insight into the relationship between sleep health and lifestyle factors. This insight could also benefit the provision of personalised recommendations for addressing individual sleep-related concerns. However, a high accuracy may also indicate that the model is overfitting the dataset.

```
#select features
#select the desired columns
selected_columns = ["Sleep Duration", "Quality of Sleep", "Physical Activity Level", "Stress Level",
                    "Daily Steps", "Age", "Systolic", "Diastolic", "GenderIndex", "OccupationIndex",
                    "BMI Status","Sleep Disorder"]

#create the new DataFrame df_selected
df_selected = df_trans4.select(selected_columns)

#verify the schema of df_selected
df_selected.printSchema()
```

```
root
 |-- Sleep Duration: float (nullable = true)
 |-- Quality of Sleep: integer (nullable = true)
 |-- Physical Activity Level: integer (nullable = true)
 |-- Stress Level: integer (nullable = true)
 |-- Daily Steps: integer (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Systolic: integer (nullable = true)
 |-- Diastolic: integer (nullable = true)
 |-- GenderIndex: double (nullable = false)
 |-- OccupationIndex: double (nullable = false)
 |-- BMI Status: integer (nullable = false)
 |-- Sleep Disorder: integer (nullable = false)
```

Figure 74: addition of select features

```
# Evaluate the model using BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator(labelCol='Sleep Disorder', metricName='areaUnderROC')
auc = evaluator.evaluate(predictions)
print(f"AUC: {auc}")
```

```
AUC: 0.9782913165266106
```

Figure 75: AUC after re-iteration

```
# Calculate the evaluation metric
accuracy_score = evaluator_multiclass.evaluate(predictions)
print('The Single Decision Tree Model has an accuracy of: {0:2.2f}%'.format(accuracy_score*100))
```

```
The Single Decision Tree Model has an accuracy of: 97.37%
```

Figure 76: Accuracy after re-iteration

```
#pattern 2: view tree model
tree_model_str = dt_model.toDebugString
print(tree_model_str)
```

```
DecisionTreeClassificationModel: uid=DecisionTreeClassifier_aa3dafe67a9d, depth=2, numNodes=7, numClasses=2, numFeatu
res=11
  If (feature 10 <= 0.5)
   If (feature 6 <= 128.5)
    Predict: 0.0
   Else (feature 6 > 128.5)
    Predict: 1.0
  Else (feature 10 > 0.5)
   If (feature 1 <= 5.5)
    Predict: 1.0
   Else (feature 1 > 5.5)
    Predict: 0.0
```

Figure 77: Decision Tree representation after re-iteration

```python
#Decision Tree presentation
for index, feature in enumerate(feature_columns):
    print(f'{index}: {feature}')
```

```
0: Sleep Duration
1: Quality of Sleep
2: Physical Activity Level
3: Stress Level
4: Daily Steps
5: Age
6: Systolic
7: Diastolic
8: GenderIndex
9: OccupationIndex
10: BMI Status
```

Figure 78: Decision tree feature detail

Bibliography

Bhardwaj, A. (2020, May 27). *Silhouette Coefficient.* Retrieved from Towards Data Science: https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c#:~:text=Silhouette%20Coefficient%20or%20silhouette%20score%20is%20a%20metric%20used%20to,each%20other%20and%20clearly%20distinguished.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45, 5–32 .

Cloud, S. (2023, July 06). *What is the C parameter in sklearn Logistic Regression?* Retrieved from Saturn Cloud: https://saturncloud.io/blog/what-is-the-c-parameter-in-sklearn-logistic-regression/#:~:text=A%20smaller%20value%20of%20C%20results%20in%20stronger%20regularization%2C%20which,and%20more%20prone%20to%20overfitting.

Clustering. (2021, Mar 01). Retrieved from IBM: https://www.ibm.com/docs/en/db2/10.1.0?topic=algorithms-clustering

Euclidean Distance . (2016). Retrieved from Science Direct: https://www.sciencedirect.com/topics/engineering/euclidean-distance

Feature Selection node. (2023, Jul 07). Retrieved from IBM: https://www.ibm.com/docs/en/cloud-paks/cp-data/4.7.x?topic=modeling-feature-selection-node

Gholamy, A., Kreinovich, V., & Kosheleva, O. (2018). *Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation.* University of Texas at El Paso .

How to Find the Best Value of C in Logistic Regression. (2023, July 10). Retrieved from SaturnCloud: https://saturncloud.io/blog/how-to-find-the-best-value-of-c-in-logistic-regression/#:~:text=L1%20regularization%20adds%20a%20penalty,controlled%20by%20the%20hyperparameter%20C.

Lee, C., & Sibley, C. (December 2019). Sleep duration and psychological well-being among New Zealanders. *ScienceDirect*, 5(6), Pages 606-614.

Li, C., & Shang, S. (Aug 2021). Relationship between Sleep and Hypertension: Findings from the NHANES (2007–2014). *International Journal of Environmental Research and Public Health*, 18(15): 7867.

Mithrakumar, M. (2019, November 12). *How to tune a Decision Tree?* . Retrieved from Towards Data Science: https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680#:~:text=The%20theoretical%20maximum%20depth%20a,one%20big%20reason%20being%20overfitting.

Modeling. (2021, Apr 09). Retrieved from IBM: https://www.ibm.com/docs/en/cloud-paks/cp-data/3.5.0?topic=palette-modeling

Patel , D., Modi , R., & Sarvakar, K. (2014, September). A Comparative Study of Clustering Data Mining: Techniques and Research Challenges. *IJLTEMAS, Volume III, Issue IX*, pp. 67-70.

Penzel, T., Kantelhardt , J. W., Lo , C.-C., Voigt, K., & Vogelmeier , C. (2003). Dynamics of Heart Rate and Sleep Stages in Normals and Patients with Sleep Apnea. *Neuropsychopharmacology*, 28, pagesS48–S53.

Rajendiran, S. (2015). *Learning classification algorithms in data mining.* Sacramento: Master of Science (MS); Computer Science; California State University.

Random Forest. (2014). Retrieved from ScienceDirect: https://www.sciencedirect.com/topics/engineering/random-forest

Roshan, J. V. (2022). Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal 15*, 531–538.

Sarker, I. H. (2021, March 22). Machine Learning: Algorithms, Real-World Applications and Research Directions. *Computer Science volume*, p. 160.

Scikit Learn. (n.d.). Retrieved from sklearn.linear_model.LogisticRegression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

sklearn.cluster.AgglomerativeClustering. (n.d.). Retrieved from Scikit Learn: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

Tokuc, A. A. (2023, May 12). *Splitting a Dataset into Train and Test Sets*. Retrieved from Bealdung: https://www.baeldung.com/cs/train-test-datasets-ratio#:~:text=With%20datasets%20containing%20considerably%20high,with%20an%2080%3A20%20split.

Vgontzas , A. N., & Fernandez-Mendoza, J. (2013). Insomnia With Short Sleep Duration. *Sleep Medicine Clinics*, 8(3), 309-322.

What is a Decision Tree. (n.d.). Retrieved from IBM: https://www.ibm.com/topics/decision-trees

What is linear regression. (n.d.). Retrieved from IBM: https://www.ibm.com/topics/linear-regression

Disclaimer