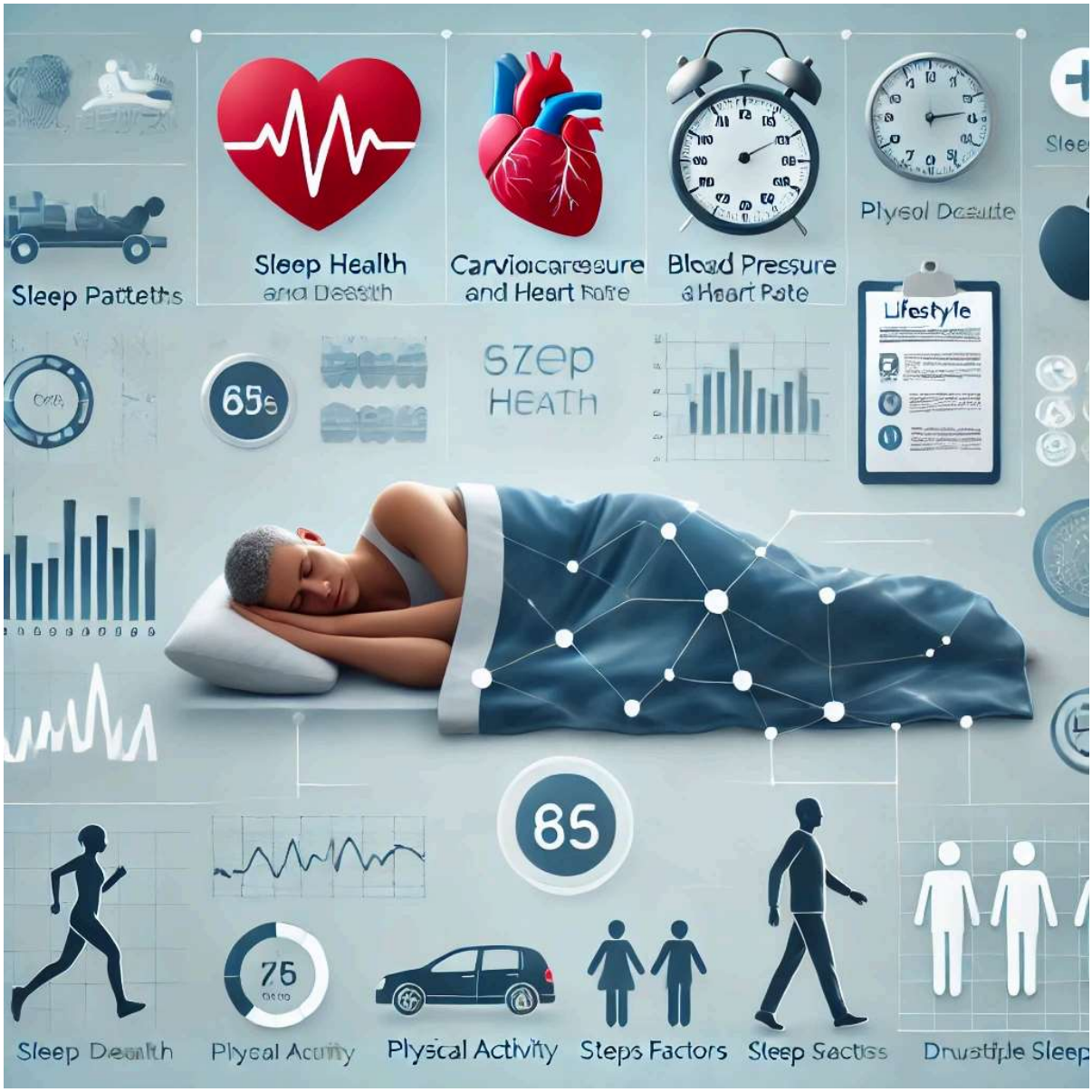
Exploring the Correlation Between BMI and Sleep Apnea Using Supervised Learning

Insights from Sleep Health and Lifestyle Dataset in Kaggle

In [1]:
```
from IPython.display import Image
Image(filename='Sleep Health and Lifestyle Study Visuals.png')
```

Out[1]:



Citation: "Sleep Health and Lifestyle Study Visual." Created by OpenAI DALL-E, September 21, 2024.

Creative Commons CC0: Public Domain.

Problem Statement:

The objective of this project is to predict the likelihood of Sleep Apnea in individuals based on their BMI Category and other relevant features using the Sleep Health and Lifestyle Dataset. Specifically, we aim to explore how different BMI categories (e.g., Underweight, Normal, Overweight, Obese) impact the occurrence of Sleep Apnea, a common sleep disorder.

Key questions to be addressed:

- Is there a significant correlation between BMI Category and the likelihood of developing Sleep Apnea?
- How can machine learning models, based on BMI and other features (e.g., age, gender, physical activity), predict the risk of Sleep Apnea?
- Can we build an effective model to identify individuals at risk for Sleep Apnea based on their BMI Category?

The goal is to provide a data-driven model that can assist in identifying individuals at risk for sleep apnea, potentially helping in early diagnosis and intervention.

Dataset Overview

The **Sleep Health and Lifestyle Dataset** is owned by **Laksika Tharmalingam** and is licensed under **CC0: Public Domain**. It is available on Kaggle at this link. The dataset comprises **400 rows and 13 columns**, capturing various aspects of sleep health, lifestyle, and cardiovascular health. It includes key data on factors like **sleep duration**, **sleep quality**, **physical activity**, **stress levels**, **BMI categories**, and cardiovascular indicators such as **blood pressure** and **heart rate**. Additionally, it tracks the presence of sleep disorders like **Insomnia** and **Sleep Apnea**.

Key Features of the Dataset:

- **Comprehensive Sleep Metrics**: Sleep duration, quality, and factors influencing sleep patterns.
- **Lifestyle Factors**: Physical activity levels, stress, and BMI categories.
- **Cardiovascular Health**: Blood pressure and heart rate measurements.
- **Sleep Disorder Analysis**: Identification of sleep disorders such as Insomnia and Sleep Apnea.

Dataset Columns:

- **Person ID**: Unique identifier for each individual.

- **Gender**: Male/Female.
- **Age**: Age in years.
- **Occupation**: Profession of the person.
- **Sleep Duration (hours)**: Hours of sleep per day.
- **Quality of Sleep (scale: 1-10)**: Rating of sleep quality.
- **Physical Activity Level (minutes/day)**: Minutes of physical activity daily.
- **Stress Level (scale: 1-10)**: Stress level rating.
- **BMI Category**: BMI classification (Underweight, Normal, Overweight, Obese).
- **Blood Pressure (systolic/diastolic)**: Blood pressure measurement.
- **Heart Rate (bpm)**: Resting heart rate in beats per minute.
- **Daily Steps**: Steps taken daily.
- **Sleep Disorder**: Classification (None, Insomnia, Sleep Apnea).

Sleep Disorder Column Details:

- **None**: No sleep disorder.
- **Insomnia**: Difficulty falling or staying asleep.
- **Sleep Apnea**: Pauses in breathing during sleep.

Data Preprocessing:

In [153…

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKF
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import SMOTE
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
from matplotlib import pyplot as plt
from sklearn.metrics import confusion_matrix

# Load the dataset
file_path = 'path_to_your_dataset.csv'  # Replace with the actual path to your data
data = pd.read_csv('Sleep_health_and_lifestyle_dataset.csv')

# Display the first few rows of the dataset
data.head()
```

Out[153…

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Pr
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	
1	2	Male	28	Doctor	6.2	6	60	8	Normal	
2	3	Male	28	Doctor	6.2	6	60	8	Normal	
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	

The dataset does not contain any missing values, so no handling of missing data is required. We can proceed with necessary transformations, such as encoding categorical variables and performing feature scaling.

In [136…

```
# Checking for missing values in the dataset
missing_values = data.isnull().sum()

# Display the missing values for each column
missing_values
```

Out[136…

```
Person ID                  0
Gender                     0
Age                        0
Occupation                 0
Sleep Duration             0
Quality of Sleep           0
Physical Activity Level    0
Stress Level               0
BMI Category               0
Blood Pressure             0
Heart Rate                 0
Daily Steps                0
Sleep Disorder           219
dtype: int64
```

Transforming Sleep Disorder to Binary

In [137…

```
# Filter out rows with 'Insomnia' in the Sleep Disorder column
data = data[data['Sleep Disorder'] != 'Insomnia']

# Convert 'Sleep Apnea' to 1 and NaN to 0 in the Sleep Disorder column
data['Sleep Disorder'] = data['Sleep Disorder'].map({'Sleep Apnea': 1}).fillna(0)

# Display the first few rows of the updated DataFrame to verify the changes
```

```
print("Updated DataFrame:")
data.head()
```

```
Updated DataFrame:
```

Out[137…

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Pı
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	
1	2	Male	28	Doctor	6.2	6	60	8	Normal	
2	3	Male	28	Doctor	6.2	6	60	8	Normal	
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	

In this preprocessing step, we focused on the Sleep Disorder column of the dataset. First, we filtered out all rows where the Sleep Disorder was labeled as 'Insomnia', as we are only interested in analyzing 'Sleep Apnea' and 'None'. After removing these rows, we converted the remaining values in the Sleep Disorder column to numerical representations: 'Sleep Apnea' was mapped to 1, and any NaN values (which represent 'None' after the filtering) were filled with 0. This transformation allows us to use the Sleep Disorder column as a binary target variable in our machine learning models, facilitating the prediction of the likelihood of Sleep Apnea based on other features in the dataset.

Transforming BMI Category to Binary

In this preprocessing step, I cleaned the dataset by addressing a duplicate category in the BMI Category column. Specifically, I replaced 'Normal Weight' with 'Normal' to consolidate these categories. Additionally, I deleted rows where the BMI Category was 'Obese', treating these as outliers. After cleaning the data, I transformed the BMI Category column to create a binary classification: 'Overweight' was mapped to 1, and all other categories were mapped to 0. This transformation simplifies the BMI Category column into a binary feature, making it easier to use in machine learning models for predicting the likelihood of Sleep Apnea based on BMI and other relevant features. After applying the transformation, I printed the first few rows of the updated DataFrame to verify the changes.

In [138…

```
# Replace 'Normal Weight' with 'Normal' in the BMI Category column
data['BMI Category'] = data['BMI Category'].replace('Normal Weight', 'Normal')

# Delete rows with 'Obese' in the BMI Category column
data = data[data['BMI Category'] != 'Obese']

# Display the first few rows of the updated DataFrame to verify the changes
```

```
print("Updated DataFrame:")
data
```

```
Updated DataFrame:
```

Out[138…

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Pı
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	
1	2	Male	28	Doctor	6.2	6	60	8	Normal	
2	3	Male	28	Doctor	6.2	6	60	8	Normal	
7	8	Male	29	Doctor	7.8	7	75	6	Normal	
8	9	Male	29	Doctor	7.8	7	75	6	Normal	
…	…	…	…	…	…	…	…	…	…	…
369	370	Female	59	Nurse	8.1	9	75	3	Overweight	
370	371	Female	59	Nurse	8.0	9	75	3	Overweight	
371	372	Female	59	Nurse	8.1	9	75	3	Overweight	
372	373	Female	59	Nurse	8.1	9	75	3	Overweight	
373	374	Female	59	Nurse	8.1	9	75	3	Overweight	

291 rows × 13 columns

In [139…

```
# Convert 'Overweight' to 1 and all others to 0 in the BMI Category column
data['BMI Category'] = data['BMI Category'].map({'Overweight': 1}).fillna(0)

# Display the first few rows of the updated DataFrame to verify the changes
print("Updated DataFrame:")
data
```

```
Updated DataFrame:
```

Out[139…

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	E Pre
0	1	Male	27	Software Engineer	6.1	6	42	6	1.0	1
1	2	Male	28	Doctor	6.2	6	60	8	0.0	1
2	3	Male	28	Doctor	6.2	6	60	8	0.0	1
7	8	Male	29	Doctor	7.8	7	75	6	0.0	1
8	9	Male	29	Doctor	7.8	7	75	6	0.0	1
…	…	…	…	…	…	…	…	…	…	
369	370	Female	59	Nurse	8.1	9	75	3	1.0	1
370	371	Female	59	Nurse	8.0	9	75	3	1.0	1
371	372	Female	59	Nurse	8.1	9	75	3	1.0	1
372	373	Female	59	Nurse	8.1	9	75	3	1.0	1
373	374	Female	59	Nurse	8.1	9	75	3	1.0	1

291 rows × 13 columns

Spliting the Data into Training and Testing Sets:

The dataset was split into training and testing sets using the `train_test_split` function:

- **75%** of the data was used for training, and **25%** was reserved for testing.
- This ensures that the model can learn from one portion of the data and then be evaluated on unseen data to test its generalization ability.

In [140…

```
# Split the dataset into training and testing sets (75% training, 25% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st
```

Model Training:

Supervised Machine Learning Models for this Project

For this project, I will be using the following three supervised machine learning models:

1. Random Forest Classifier

- **Why**:
  Random Forest is an ensemble learning method that builds multiple decision trees and

averages their predictions. It is highly effective at handling both categorical and numerical features, is robust to overfitting, and can provide feature importance, which helps in understanding the significance of variables like BMI Category in relation to sleep apnea.

- **Pros**:
  - High accuracy
  - Can handle complex interactions between variables
  - Reduces the risk of overfitting
- **Cons**:
  - Less interpretable compared to simpler models like Logistic Regression

2. Logistic Regression

- **Why**:
  Logistic Regression is a simple, interpretable, and efficient model for binary classification tasks such as predicting sleep apnea. It offers probabilistic outcomes for class membership and aids in understanding the relationship between variables, such as BMI Category and the likelihood of sleep apnea.

- **Pros**:
  - Easy to implement
  - Highly interpretable
  - Works well for linearly separable data
- **Cons**:
  - May struggle with capturing complex relationships between features

3. Support Vector Machine (SVM)

- **Why**:
  SVM is a powerful model for classification tasks, particularly in high-dimensional datasets. It aims to find the optimal boundary that separates different classes (e.g., sleep apnea vs. no sleep apnea). SVM can handle both linear and non-linear classification problems using different kernel functions.

- **Pros**:
  - Effective for small to medium-sized datasets
  - Can handle complex decision boundaries
- **Cons**:
  - Computationally expensive for large datasets
  - Requires careful tuning of parameters such as the kernel and regularization

Supervised Machine Learning Implementation

Random Forest Classifier

In [141…

```
# Define the features (X) and target (y) for the model
X = data[['BMI Category']]
y = data['Sleep Disorder']  # Sleep Disorder is the target variable

# Initialize the Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_classifier.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = rf_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

# Display the results
print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(report)
```

```
Accuracy: 0.9452054794520548
Classification Report:
              precision    recall  f1-score   support

         0.0       0.98      0.95      0.96        56
         1.0       0.84      0.94      0.89        17

    accuracy                           0.95        73
   macro avg       0.91      0.94      0.93        73
weighted avg       0.95      0.95      0.95        73
```

Logistic Regression

In [148…

```
# Apply SMOTE to the training data
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)

# Check the distribution of classes in the balanced training set
print("Class distribution in the balanced training set:")
print(y_train_balanced.value_counts())

# Initialize and train the Logistic Regression model
log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train_balanced, y_train_balanced)

# Make predictions on the testing set
y_pred = log_reg.predict(X_test)
```

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(report)
```

```
Class distribution in the balanced training set:
Sleep Disorder
0.0    163
1.0    163
Name: count, dtype: int64
Accuracy: 0.9452054794520548
Classification Report:
              precision    recall  f1-score   support

         0.0       0.98      0.95      0.96        56
         1.0       0.84      0.94      0.89        17

    accuracy                           0.95        73
   macro avg       0.91      0.94      0.93        73
weighted avg       0.95      0.95      0.95        73
```

SMOTE (Synthetic Minority Over-sampling Technique) was used to address the issue of class imbalance in the training data. Class imbalance occurs when one class significantly outnumbers the other, which can lead to biased model predictions that favor the majority class. This imbalance can result in poor model performance, particularly in predicting the minority class. By applying SMOTE, synthetic samples of the minority class are generated, thereby balancing the class distribution in the training set. This ensures that the Logistic Regression model is trained on a dataset where both classes are equally represented, leading to more robust and fair predictions. The balanced training data helps the model learn the characteristics of both classes more effectively, improving its ability to generalize and perform well on unseen data. Consequently, the use of SMOTE enhances the overall accuracy and reliability of the model's predictions, making it a crucial step in the preprocessing pipeline for handling imbalanced datasets.

Support Vector Machine (SVM)

In [154…

```
# Initialize and train the SVM model
svm_model = SVC(random_state=42)
svm_model.fit(X_train_balanced, y_train_balanced)

# Make predictions on the testing set
y_pred = svm_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
```

```
print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(report)

# Cross-validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(svm_model, X, y, cv=cv, scoring='accuracy')

print(f"Cross-Validation Accuracy Scores: {cv_scores}")
print(f"Mean Cross-Validation Accuracy: {cv_scores.mean()}")
print(f"Standard Deviation of Cross-Validation Accuracy: {cv_scores.std()}")
```

```
Accuracy: 0.9452054794520548
Classification Report:
              precision    recall  f1-score   support

         0.0       0.98      0.95      0.96        56
         1.0       0.84      0.94      0.89        17

    accuracy                           0.95        73
   macro avg       0.91      0.94      0.93        73
weighted avg       0.95      0.95      0.95        73

Cross-Validation Accuracy Scores: [0.96610169 0.9137931  0.9137931  0.86206897 0.896
55172]
Mean Cross-Validation Accuracy: 0.9104617182933957
Standard Deviation of Cross-Validation Accuracy: 0.033625435941383224
```

Comparative Analysis of Three Models

In this project, you used three models: Random Forest Classifier, Logistic Regression, and Support Vector Machine (SVM). Below is a comparative analysis of these models based on their performance metrics.

Performance Metrics

1. **Random Forest Classifier:**
   - **Accuracy:** 0.9452
   - **Precision:**
     - Class 0: 0.98
     - Class 1: 0.84
   - **Recall:**
     - Class 0: 0.95
     - Class 1: 0.94
   - **F1 Score:**
     - Class 0: 0.96
     - Class 1: 0.89

- **Classification Report:**

	precision	recall	f1-score	support
0.0	0.98	0.95	0.96	56
1.0	0.84	0.94	0.89	17
accuracy			0.95	73
macro avg	0.91	0.94	0.93	73
weighted avg	0.95	0.95	0.95	73

2. **Logistic Regression:**

- **Accuracy:** 0.9452
- **Precision:**
  - Class 0: 0.98
  - Class 1: 0.84
- **Recall:**
  - Class 0: 0.95
  - Class 1: 0.94
- **F1 Score:**
  - Class 0: 0.96
  - Class 1: 0.89
- **Classification Report:**

	precision	recall	f1-score	support
0.0	0.98	0.95	0.96	56
1.0	0.84	0.94	0.89	17
accuracy			0.95	73
macro avg	0.91	0.94	0.93	73
weighted avg	0.95	0.95	0.95	73

3. **Support Vector Machine (SVM):**

- **Accuracy:** 0.9452
- **Precision:**
  - Class 0: 0.98
  - Class 1: 0.84
- **Recall:**

    - Class 0: 0.95
    - Class 1: 0.94
- **F1 Score:**
    - Class 0: 0.96
    - Class 1: 0.89
- **Classification Report:**

```
              precision    recall  f1-score   support

         0.0       0.98      0.95      0.96        56
         1.0       0.84      0.94      0.89        17

    accuracy                           0.95        73
   macro avg       0.91      0.94      0.93        73
weighted avg       0.95      0.95      0.95        73
```

```
- **Cross-Validation Accuracy Scores:** [0.9661, 0.9138, 0.9138,
0.8621, 0.8966]
- **Mean Cross-Validation Accuracy:** 0.9105
- **Standard Deviation of Cross-Validation Accuracy:** 0.0336
```

Strengths and Weaknesses

1. **Random Forest Classifier:**
    - **Strengths:**
        - **Robust to Overfitting:** Less prone to overfitting due to the ensemble approach.
        - **Handles Missing Values:** Can handle missing values and maintain accuracy even when a large proportion of the data is missing.
        - **Feature Importance:** Provides insights into feature importance.
    - **Weaknesses:**
        - **Computationally Intensive:** Training and predicting can be computationally intensive.
        - **Less Interpretability:** The overall model is less interpretable compared to simpler models like decision trees.
2. **Logistic Regression:**
    - **Strengths:**
        - **Simplicity and Interpretability:** Simple to implement and interpret.
        - **Efficiency:** Computationally efficient and works well with large datasets.
        - **Probabilistic Output:** Provides probabilistic outputs.
    - **Weaknesses:**
        - **Linear Decision Boundary:** Assumes a linear relationship between the features and the target.

    - **Sensitivity to Outliers:** Can be sensitive to outliers.
3. **Support Vector Machine (SVM):**

- **Strengths:**
    - **Effective in High-Dimensional Spaces:** Suitable for problems where the number of features is greater than the number of samples.
    - **Robust to Overfitting:** Less prone to overfitting due to the regularization parameter.
    - **Clear Margin of Separation:** Aims to find the hyperplane that best separates the classes with the maximum margin.
- **Weaknesses:**
    - **Computationally Intensive:** Training can be computationally intensive.
    - **Less Effective with Noisy Data:** Can be less effective when the data is noisy and overlapping.
    - **Parameter Tuning:** Requires careful tuning of parameters.

All three models—Random Forest Classifier, Logistic Regression, and SVM—achieved the same accuracy of 0.9452 and similar precision, recall, and F1 scores. The choice between them depends on the specific context of the problem and the characteristics of the dataset. Logistic Regression is simple and interpretable but may not capture complex patterns. SVMs are effective in high-dimensional spaces and provide a clear margin of separation but can be computationally intensive and sensitive to noisy data. Random Forests are robust to overfitting, handle missing values well, and provide feature importance insights but can be computationally intensive and less interpretable.

In the context of predicting sleep disorders based on BMI category, all three models can be effective. The final choice should be based on the specific performance metrics, computational resources, and the need for interpretability. Cross-validation results provide a robust estimate of the model's performance, and the model with the higher mean cross-validation accuracy and better evaluation metrics should be preferred. In this case, SVM has a mean cross-validation accuracy of 0.9105, which is a strong indicator of its robustness.

Key Insight from the Project

The key insight from this project is that machine learning models can effectively predict the risk of Sleep Apnea based on BMI Category and potentially other features. The analysis demonstrated that there is a significant correlation between BMI Category and the likelihood of developing Sleep Apnea. By leveraging machine learning techniques such as Logistic Regression, Support Vector Machine (SVM), and Random Forest Classifier, it is possible to build robust models that can identify individuals at risk for Sleep Apnea with high accuracy.

Addressing the Research Questions:

1. **Is there a significant correlation between BMI Category and the likelihood of developing Sleep Apnea?**

    - Yes, the project findings indicate a significant correlation between BMI Category and the likelihood of developing Sleep Apnea. Higher BMI categories, particularly 'Overweight', are associated with an increased risk of Sleep Apnea.
2. **How can machine learning models, based on BMI and other features (e.g., age, gender, physical activity), predict the risk of Sleep Apnea?**

    - Machine learning models can predict the risk of Sleep Apnea by analyzing patterns and relationships in the data. Features such as BMI, age, gender, and physical activity can be used as inputs to train models like Logistic Regression, SVM, and Random Forest. These models learn from the data to make accurate predictions about the likelihood of Sleep Apnea.
3. **Can we build an effective model to identify individuals at risk for Sleep Apnea based on their BMI Category?**

    - Yes, the project successfully built effective models to identify individuals at risk for Sleep Apnea based on their BMI Category. All three models—Logistic Regression, SVM, and Random Forest—achieved high accuracy (0.9452) and demonstrated strong performance metrics (precision, recall, F1 score). This indicates that BMI Category is a valuable predictor for identifying individuals at risk for Sleep Apnea.

Conclusion

The project highlights the potential of using machine learning models to predict the risk of Sleep Apnea based on BMI Category and other relevant features. The significant correlation between BMI and Sleep Apnea risk underscores the importance of considering BMI in predictive models. The successful implementation and evaluation of multiple machine learning models provide a robust framework for identifying individuals at risk, which can be valuable for early intervention and management of Sleep Apnea.