Model training based on sleep health and lifestyle dataset

Course: CPS844 Data Mining

Term : winter 25

Prof : Chen (Cherie) Ding

Team members:

041112129 Yan huang

501098874 Ziwei Yao

Introduction

The dataset we used in this project is the Sleep Health and Lifestyle statistical survey data downloaded from Kaggle. Its attributes and brief descriptions are as follows:

Person ID: An unique identifier for each individual.

Gender: The gender of the person (Male/Female).

Age: The age of the person in years.

Occupation: The occupation or profession of the person.

Sleep Duration (hours): The number of hours the person sleeps per day.

Quality of Sleep (scale: 1-10): A subjective rating of the quality of sleep, ranging from 1 to 10.

Physical Activity Level (minutes/day): The number of minutes the person engages in physical activity daily.

Stress Level (scale: 1-10): A subjective rating of the stress level experienced by the person, ranging from 1 to 10.

BMI Category: The BMI category of the person (e.g., Underweight, Normal, Overweight).

Blood Pressure (systolic/diastolic): The blood pressure measurement of the person, indicated as systolic pressure over diastolic pressure.

Heart Rate (bpm): The resting heart rate of the person in beats per minute.

Daily Steps: The number of steps the person takes per day.

Sleep Disorder: The presence or absence of a sleep disorder in the person (None, Insomnia, Sleep Apnea).

This project is about regression, and the goal is to predict sleep quality given relevant parameters. After taking a closer look, we can see that this dataset is very clean, and it has probably gone through multiple rounds of pre-processing.

Nonetheless, we will try to do data preprocessing with feature selection, at the same time also doing sampling. Then, the qualitative data will undergo some discretization and be transformed to numerical data types. This way, we can directly use the numerical input as parameters, and do calculations as well as comparisons.

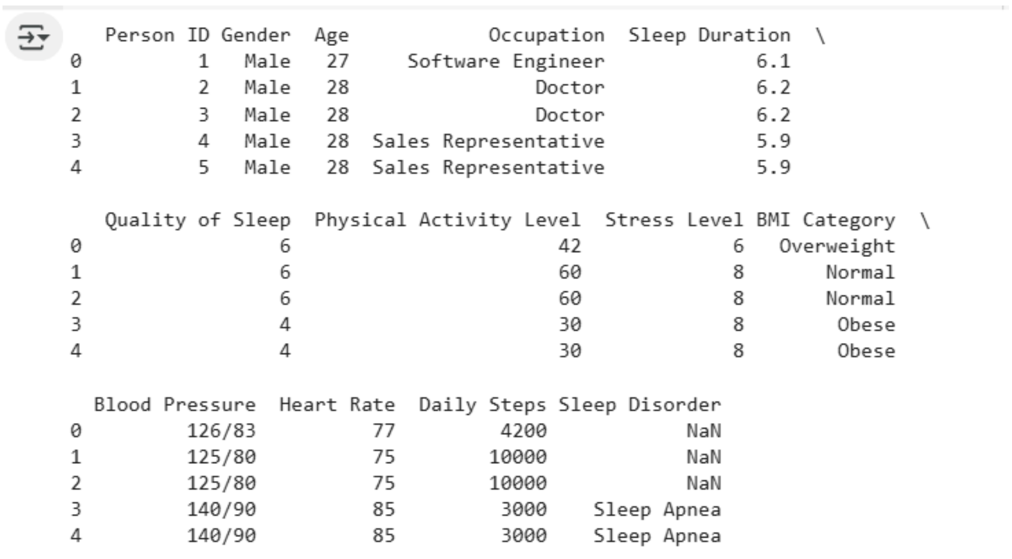
For the actual algorithms to be done, we will implement 5 different regression models, including linear, SVR, decision tree, random forest, and a feature selection (Lasso). For some of the regression models, we did more than one run with slightly modified parameters (SVR and Lasso). Then, we did cross validation so that we can see how the model performs under different conditions.

In short, after pre-processing, we are going to use 5 regression models, and compare the resulting metrics from all these runs, by using MSE (mean square error) and R2 score; as well as posting any diagrams for easier visual inspection.

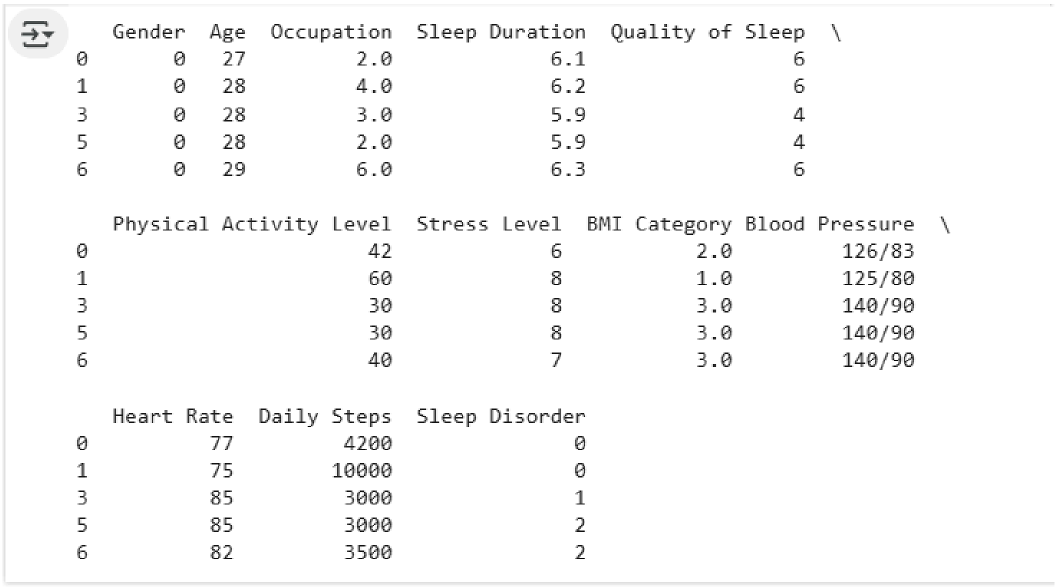
Preprocessing

Label Encoding

Before and after label encoding:



```
   Person ID Gender  Age            Occupation  Sleep Duration  \
0          1   Male   27     Software Engineer             6.1
1          2   Male   28                Doctor             6.2
2          3   Male   28                Doctor             6.2
3          4   Male   28  Sales Representative             5.9
4          5   Male   28  Sales Representative             5.9

   Quality of Sleep  Physical Activity Level  Stress Level BMI Category  \
0                 6                       42             6   Overweight
1                 6                       60             8       Normal
2                 6                       60             8       Normal
3                 4                       30             8        Obese
4                 4                       30             8        Obese

  Blood Pressure  Heart Rate  Daily Steps Sleep Disorder
0         126/83          77         4200            NaN
1         125/80          75        10000            NaN
2         125/80          75        10000            NaN
3         140/90          85         3000    Sleep Apnea
4         140/90          85         3000    Sleep Apnea
```



```
   Gender  Age  Occupation  Sleep Duration  Quality of Sleep  \
0       0   27         2.0             6.1                 6
1       0   28         4.0             6.2                 6
3       0   28         3.0             5.9                 4
5       0   28         2.0             5.9                 4
6       0   29         6.0             6.3                 6

   Physical Activity Level  Stress Level  BMI Category Blood Pressure  \
0                       42             6           2.0         126/83
1                       60             8           1.0         125/80
3                       30             8           3.0         140/90
5                       30             8           3.0         140/90
6                       40             7           3.0         140/90

   Heart Rate  Daily Steps  Sleep Disorder
0          77         4200               0
1          75        10000               0
3          85         3000               1
5          85         3000               2
6          82         3500               2
```

The preprocessing steps will start by label encoding of some of the qualitative variables, such as occupation, BMI category and sleep disorder.

Gender collected here is binary; we will use 0 and 1 to represent it.

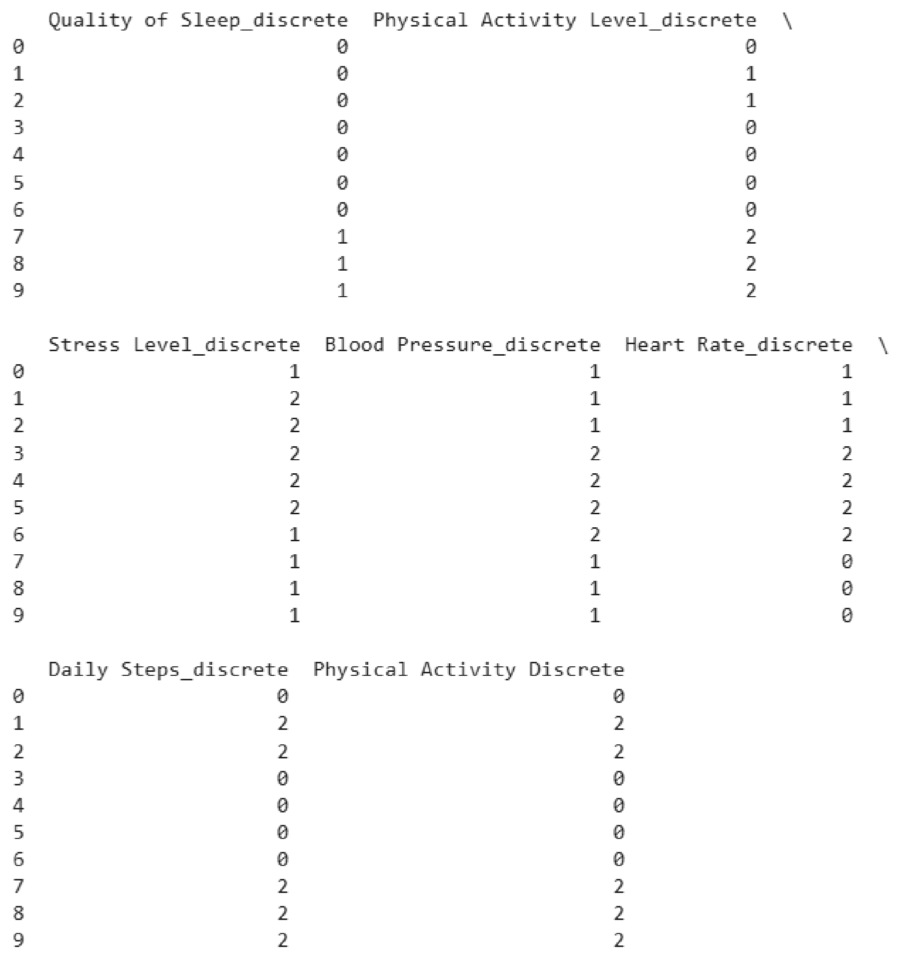
For BMI, the dataset used a qualitative measure: normal, overweight and obese. We will turn it into numerical values of 1, 2 and 3, respectively.

Occupation has slightly more variations than BMI, it includes: Engineer, Software Engineer, Sales Representative, Sales Person, Doctor, Nurse, Teacher, Accountant, Scientist, Lawyer, and Manager. We will use numbers 1 to 1 to represent these. Also, some of them may be subjected to merging as the process goes along (such as sales representatives and salespeople).

Sleep Disorder is another qualitative attribute, which only includes: None, Sleep Apnea and insomnia. These values will be transformed into 0, 1 and 2 ,respectively.

For numerical values, we did data splitting for blood pressure, since it involves two numbers. Splitting the value makes the data cleaner and easier for further analysis.

Discretization

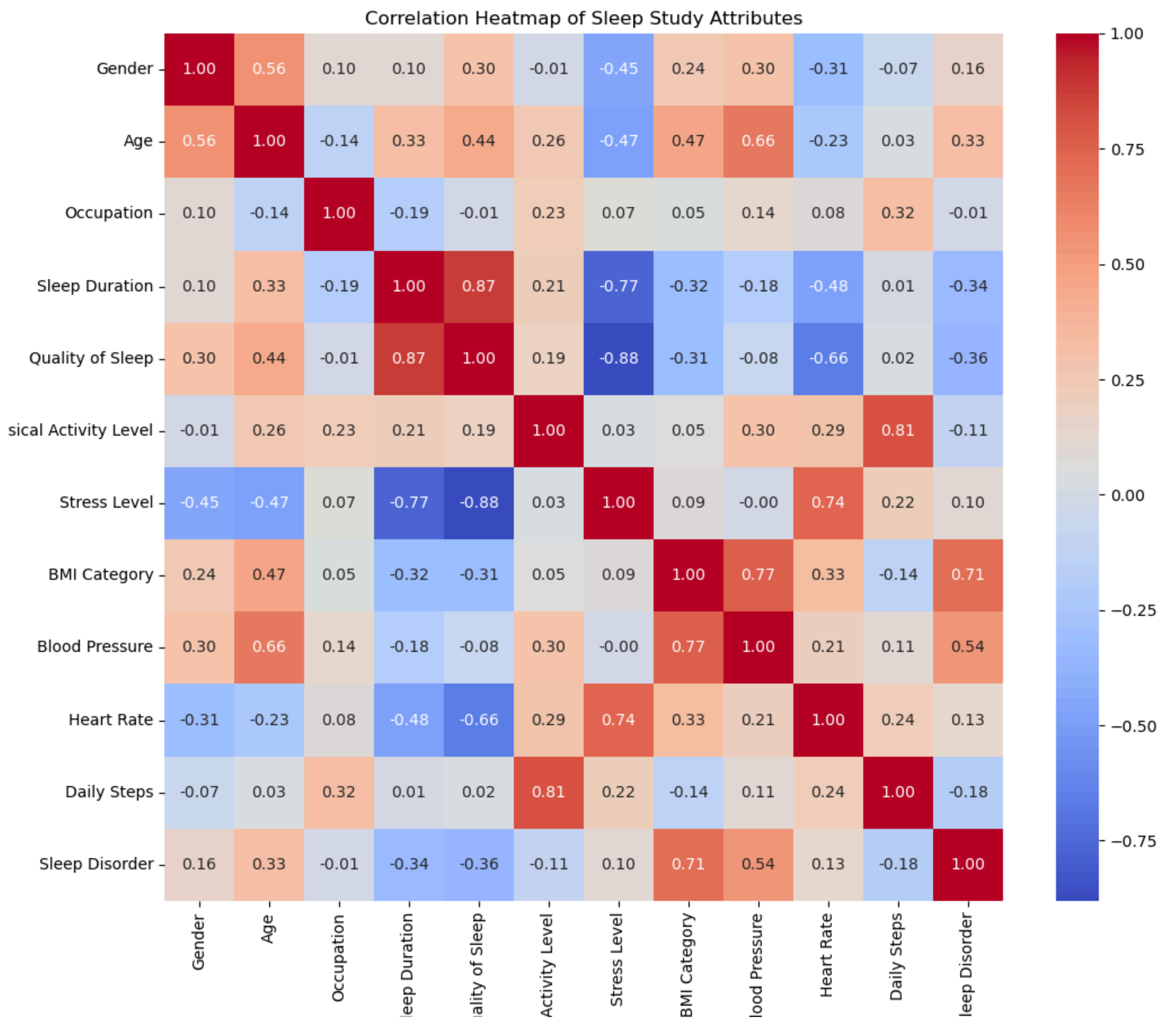


```
   Quality of Sleep_discrete  Physical Activity Level_discrete  \
0                         0                                 0
1                         0                                 1
2                         0                                 1
3                         0                                 0
4                         0                                 0
5                         0                                 0
6                         0                                 0
7                         1                                 2
8                         1                                 2
9                         1                                 2

   Stress Level_discrete  Blood Pressure_discrete  Heart Rate_discrete  \
0                      1                        1                    1
1                      2                        1                    1
2                      2                        1                    1
3                      2                        2                    2
4                      2                        2                    2
5                      2                        2                    2
6                      1                        2                    2
7                      1                        1                    0
8                      1                        1                    0
9                      1                        1                    0

   Daily Steps_discrete  Physical Activity Discrete
0                     0                           0
1                     2                           2
2                     2                           2
3                     0                           0
4                     0                           0
5                     0                           0
6                     0                           0
7                     2                           2
8                     2                           2
9                     2                           2
```

We did the discretization by dividing each attribute into bins, assigning them a new numerical value, while preserving the underlying meaning of the data. This way, the data is easier to read and the algorithms can focus on the patterns only, and not having to worry about the details. For instance, we divide sleep quality (originally 1 to 10) into 5 separate bins like: 0 (very poor), 1 (poor), 2 (average), and 5 (excellent).

For this particular project, we chose K-means as our binning/discretization method. Unlike equal-width or equal frequency binning, we can better capture the actual distribution of the data, especially when the boundaries between each data point are not clearly defined. In addition, we can reduce the risk of overfitting by setting up meaningful clusters, and our intended classes are still keeping the numerical nature of the data. In doing so, we hope to improve how the model performs so we can have a better understanding when comparing algorithms and their results.

Feature Selection



The purpose of feature selection is to select reasonable attributes for training. Reasonable feature selection will bring higher accuracy. In the course, we have learned three feature selection methods, namely embedded approaches, filter approaches, and wrapper approaches. Before using them, it is very important to know their functions. The first is the embedded approaches. When using this method, feature selection will occur in the algorithm process.

According to this description, decision tree regression will be a good example of the implementation of this method. The second method is the Filter Approach, however, this requires manual measurement. One of the methods is to directly check the variance of the data. If the variance is too small, it means that the impact and correlation of this data will not be very large (of course, if the variance of the whole dataset is not large, then it is another story). Another way is to ask humans to judge what data should not have a direct correlation to avoid coincidence.

For example, the most obvious first step is to remove the "person ID" column, because it will not have a valid association with any data. In other words, if the person ID is removed, the accuracy of the model will not be affected, and almost guaranteed to increase, so that will be implemented in every run. Another case is the blood pressure attribute, where it involves two numeric values of Systolic and Diastolic readings; due to the minimal difference among the SST in these attributes, we decided to discard partial values and keep the Systolic blood pressure only, which is suffice enough to indicate any condition of elevated blood pressure.

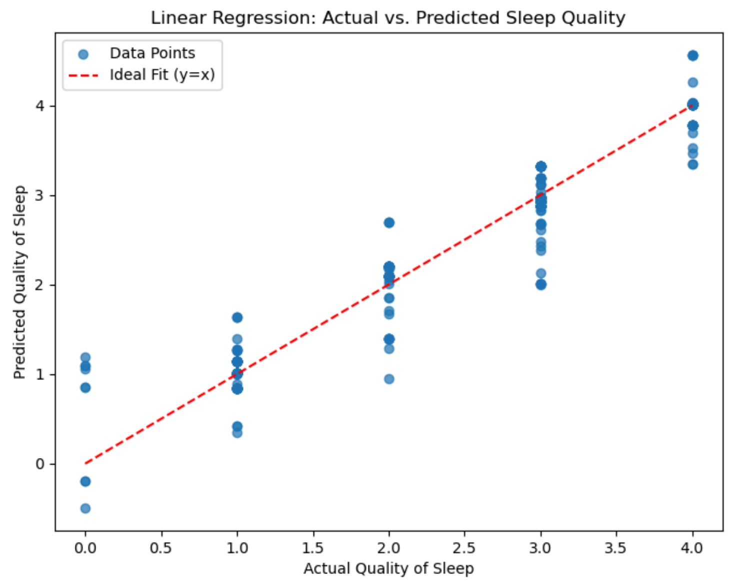
Wrapper approaches function similarly to manual exclusion. It tests by training and evaluating models iteratively, selecting features based on their impact on performance. However, when the dataset is small, it may cause overfitting. To mitigate this, we consider combining them with filter approaches to achieve a more balanced selection.

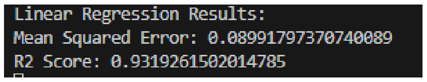
First Run：Linear Regression

For the first run, we employed linear regression, with the purpose of finding the linear relationship between sleep quality and all the other attributes after training. The final result of this algorithm is obtained by analyzing the relationships between each attribute and the target, and calculating their weights, and then combining them to produce the predicted values on the Y-axis. The actual sleep quality labels are placed on the X-axis, and these are used to train a best-fit line that serves as the prediction model.

If we plotted the weighted results from the original data, we could observe that the points align along the X-axis in the values of 0, 1, 2, 3, and 4—which are the actual possible values for sleep quality. Around these positions, we can also see darker-colored overlapping circles and the place where the predicted line intersects, which is indicating that the majority of the test data points lie very close to the regression line. This visual alignment can also be double-checked by the evaluation metrics: a Mean Squared Error of 0.0765 and an R^2 Score of 0.9479, confirming the model's strong predictive performance.

We also did another separate run, trying to use only the set of attributes that has high relevance to the sleep quality. However, we ended up with a lower value for R2. This may be because of the weighted calculation of the python library, where the low weighted attributes still contribute to the model, without worrying about them influencing the result or accuracy. Moreover, despite the attribute relevance seen in heat maps, the indicators there alone is not representative enough of the combined effect of all the attributes in a multivariate regression model, thus, removing them negatively impacts the final result and the R2 score.



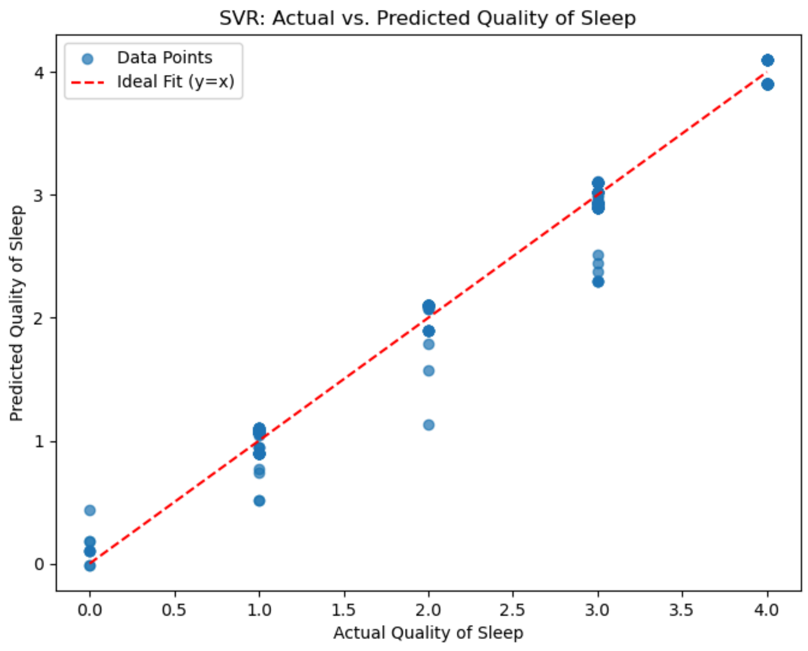


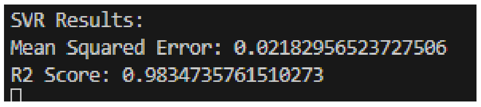
```
Linear Regression Results:
Mean Squared Error: 0.08991797370740089
R2 Score: 0.9319261502014785
```

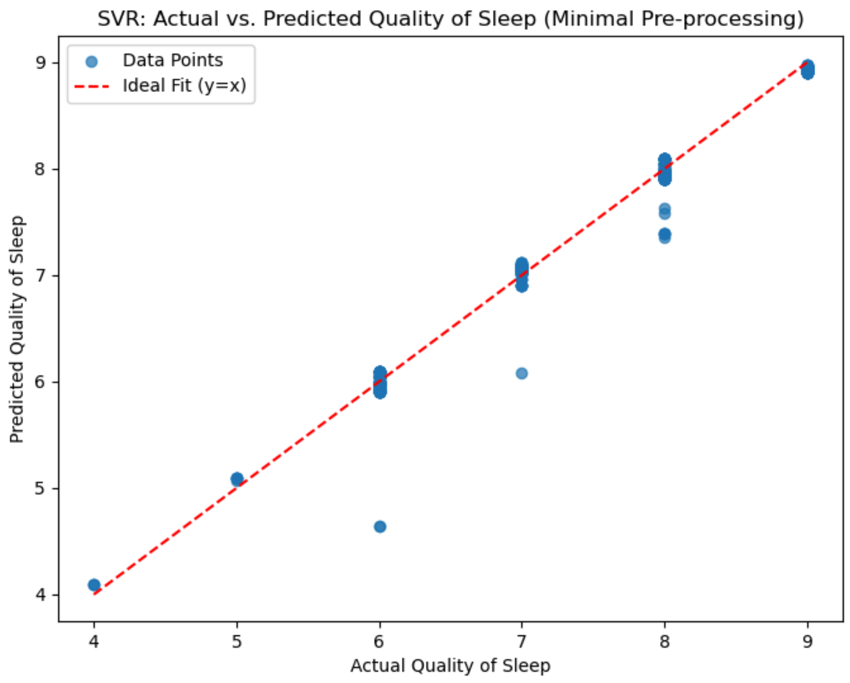
Second Run： Support Vector Regression

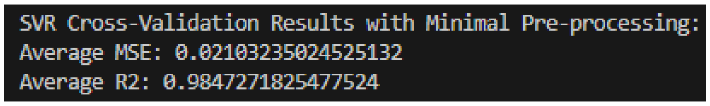
SVR is from Support Vector Machine(SVM) used for regression rather than classification. It finds a model that maps the input features to the target value. This method is mostly used in modelling non-linear relationships, in which it can project data through higher dimensions.

In practice, we conducted two separate SVR trials, one with normal pre-processing, and another one with minimal pre-processing. Since we did discretization as part of the pre-processing, the idea for the second run is to see what happens when the values are continuous, in an attempt to better preserve the continuous numeric nature of the data. Additionally, we introduced a 5-fold cross validation technique to further ensure robustness and avoid overfitting. And it turns out that the R2 score for the 2nd attempt is actually lower. The results confirm our expectation that preprocessing is critical for achieving optimal performance.




```
SVR Results:
Mean Squared Error: 0.02182956523727506
R2 Score: 0.9834735761510273
```

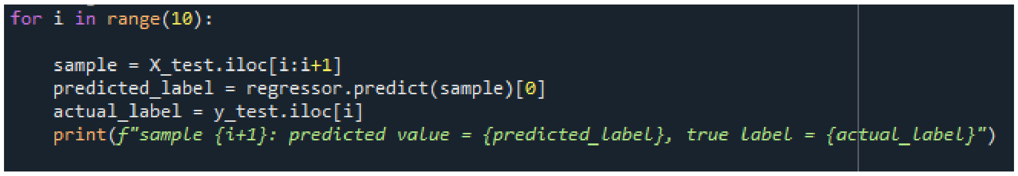




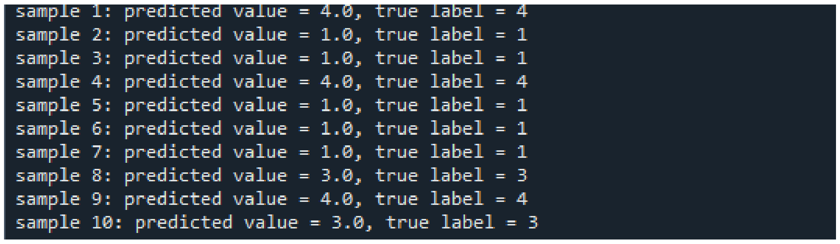
```
SVR Cross-Validation Results with Minimal Pre-processing:
Average MSE: 0.02103235024525132
Average R2: 0.9847271825477524
```

Third Run: Decision Tree Regression

Compared to linear regression and the general predictions done by its weighted calculation and training, the decision tree directly uses each point. Although harder to visualize, its structure is not so abstract, and the logic of the decision tree can be seen via graphical representation - as its name implies: a tree. Decision tree is definitely a more complicated algorithm than linear regression, rather than generating a single line (which takes just an instant), the time it took for decision tree model to reach a conclusion is almost half a minute. It uses attributes as splitting points, to train the classification needed for regression modeling. When one set of attributes are used as input, this algorithm will use part of the tree that's already built to seek a predictive value, while each split is determined by the value ranges of the attributes. The predictions can be seen at the very end of the tree. We can try to use some of the test set data to see how accurate it may be

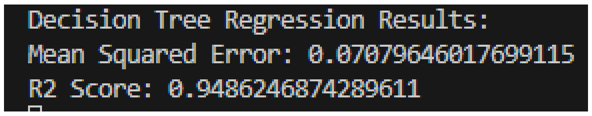


```
for i in range(10):

    sample = X_test.iloc[i:i+1]
    predicted_label = regressor.predict(sample)[0]
    actual_label = y_test.iloc[i]
    print(f"sample {i+1}: predicted value = {predicted_label}, true label = {actual_label}")
```



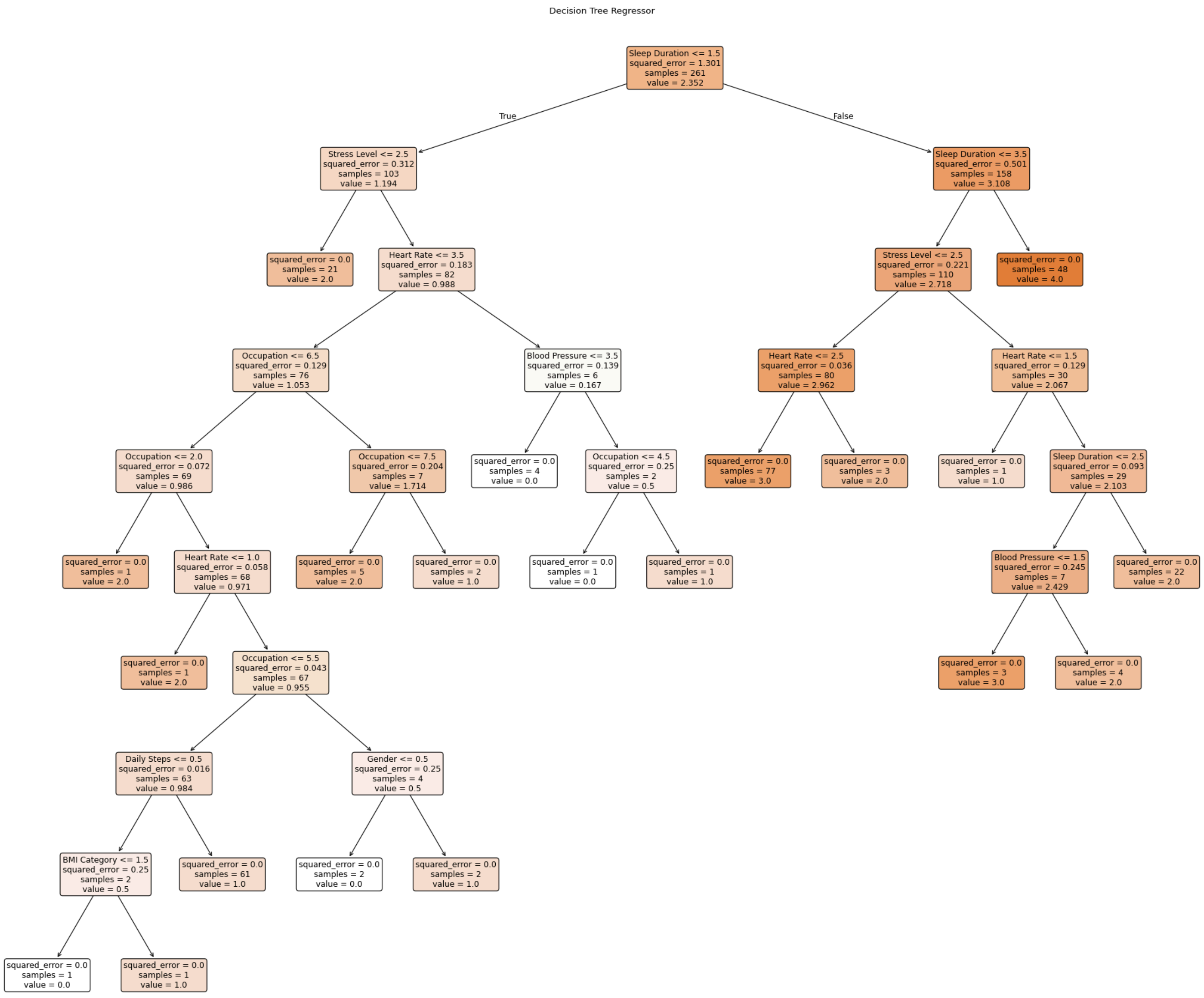
```
sample 1: predicted value = 4.0, true label = 4
sample 2: predicted value = 1.0, true label = 1
sample 3: predicted value = 1.0, true label = 1
sample 4: predicted value = 4.0, true label = 4
sample 5: predicted value = 1.0, true label = 1
sample 6: predicted value = 1.0, true label = 1
sample 7: predicted value = 1.0, true label = 1
sample 8: predicted value = 3.0, true label = 3
sample 9: predicted value = 4.0, true label = 4
sample 10: predicted value = 3.0, true label = 3
```

In here we used a for loop and range functions to choose the top 10 test data and their predictive value. We then used (predicted value == true label) as the input for np.mean to calculate the accuracy of test data, and the result is as high as 0.9486246874289611. To avoid overfitting, we selected 30% as the amount of the test dataset, if it had been 20%, then the result would have been almost 100%.



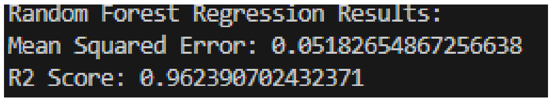
```
Decision Tree Regression Results:
Mean Squared Error: 0.07079646017699115
R2 Score: 0.9486246874289611
```

Below is the graph of the decision tree:


Decision Tree Regressor
Sleep Duration <= 1.5 squared_error = 1.301 samples = 261 value = 2.352
True
False
Stress Level <= 2.5 squared_error = 0.312 samples = 103 value = 1.194
Sleep Duration <= 3.5 squared_error = 0.501 samples = 158 value = 3.108
squared_error = 0.0 samples = 21 value = 2.0
Heart Rate <= 3.5 squared_error = 0.183 samples = 82 value = 0.988
Stress Level <= 2.5 squared_error = 0.221 samples = 110 value = 2.718
squared_error = 0.0 samples = 48 value = 4.0
Occupation <= 6.5 squared_error = 0.129 samples = 76 value = 1.053
Blood Pressure <= 3.5 squared_error = 0.139 samples = 6 value = 0.167
Heart Rate <= 2.5 squared_error = 0.036 samples = 80 value = 2.962
Heart Rate <= 1.5 squared_error = 0.129 samples = 30 value = 2.067
Occupation <= 2.0 squared_error = 0.072 samples = 69 value = 0.986
Occupation <= 7.5 squared_error = 0.204 samples = 7 value = 1.714
squared_error = 0.0 samples = 4 value = 0.0
Occupation <= 4.5 squared_error = 0.25 samples = 2 value = 0.5
squared_error = 0.0 samples = 77 value = 3.0
squared_error = 0.0 samples = 3 value = 2.0
squared_error = 0.0 samples = 1 value = 1.0
Sleep Duration <= 2.5 squared_error = 0.093 samples = 29 value = 2.103
squared_error = 0.0 samples = 1 value = 2.0
Heart Rate <= 1.0 squared_error = 0.058 samples = 68 value = 0.971
squared_error = 0.0 samples = 5 value = 2.0
squared_error = 0.0 samples = 2 value = 1.0
squared_error = 0.0 samples = 1 value = 0.0
squared_error = 0.0 samples = 1 value = 1.0
Blood Pressure <= 1.5 squared_error = 0.245 samples = 7 value = 2.429
squared_error = 0.0 samples = 22 value = 2.0
squared_error = 0.0 samples = 1 value = 2.0
Occupation <= 5.5 squared_error = 0.043 samples = 67 value = 0.955
squared_error = 0.0 samples = 3 value = 3.0
squared_error = 0.0 samples = 4 value = 2.0
Daily Steps <= 0.5 squared_error = 0.016 samples = 63 value = 0.984
Gender <= 0.5 squared_error = 0.25 samples = 4 value = 0.5
BMI Category <= 1.5 squared_error = 0.25 samples = 2 value = 0.5
squared_error = 0.0 samples = 61 value = 1.0
squared_error = 0.0 samples = 2 value = 0.0
squared_error = 0.0 samples = 2 value = 1.0
squared_error = 0.0 samples = 1 value = 0.0
squared_error = 0.0 samples = 1 value = 1.0

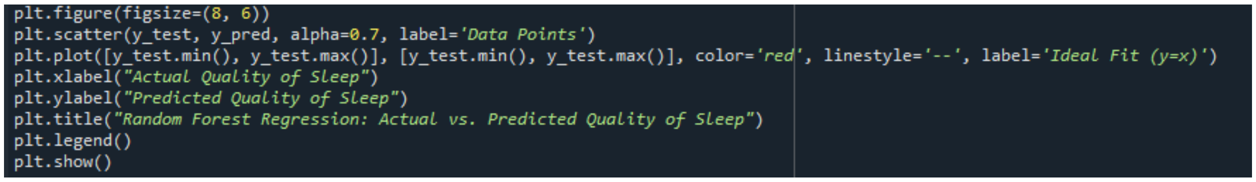
Fourth Run: Random Forest

While we were training the decision tree model, we noticed that it still had risks of being overfitting, and we needed a way to mitigate that. During our studies in sampling methods, we mentioned the algorithm of random forest, and it uses a bagging-like algorithm, repeatedly sampling the data (with replacement) to train multiple decision trees independently, then integrating them (just bagging sampling) to avoid possible overfitting. Therefore, the R2 value for random forest is even higher, because of the lower tendency to overfit which makes it more suitable for predicting test data.

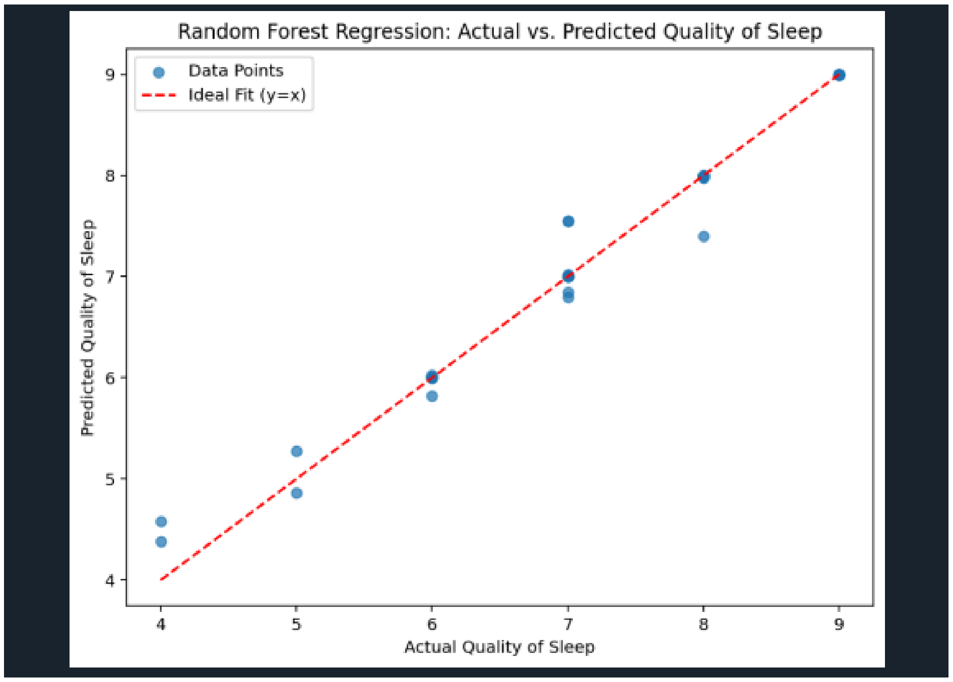


```
Random Forest Regression Results:
Mean Squared Error: 0.05182654867256638
R2 Score: 0.962390702432371
```

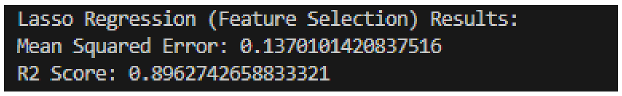
In the end, we used the plt library to generate a comparison between predicted value and actual value, we can see that the difference between them is rather minimal.



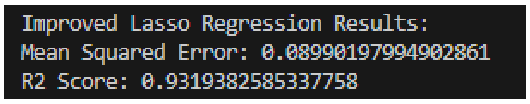
```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.7, label='Data Points')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', linestyle='--', label='Ideal Fit (y=x)')
plt.xlabel("Actual Quality of Sleep")
plt.ylabel("Predicted Quality of Sleep")
plt.title("Random Forest Regression: Actual vs. Predicted Quality of Sleep")
plt.legend()
plt.show()
```



Fifth Run: Lasso Regression



```
Lasso Regression (Feature Selection) Results:
Mean Squared Error: 0.1370101420837516
R2 Score: 0.8962742658833321
```



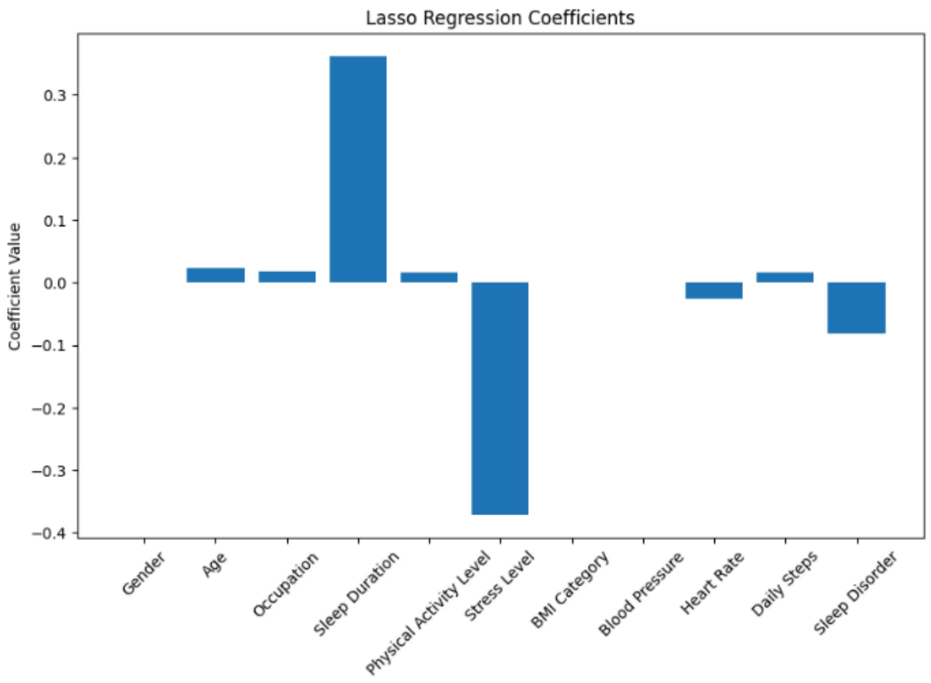
```
Improved Lasso Regression Results:
Mean Squared Error: 0.08990197994902861
R2 Score: 0.9319382585337758
```

We chose Lasso regression as our last because it is a classic feature selection algorithm. Here, we actually did Lasso twice—the second time we did it with GridSearchCV. Lasso regression does feature selection by penalizing the less important features, thus effectively eliminating some attributes by making their coefficients to zero. This makes it especially useful for identifying which attributes actually contribute to predicting sleep quality.

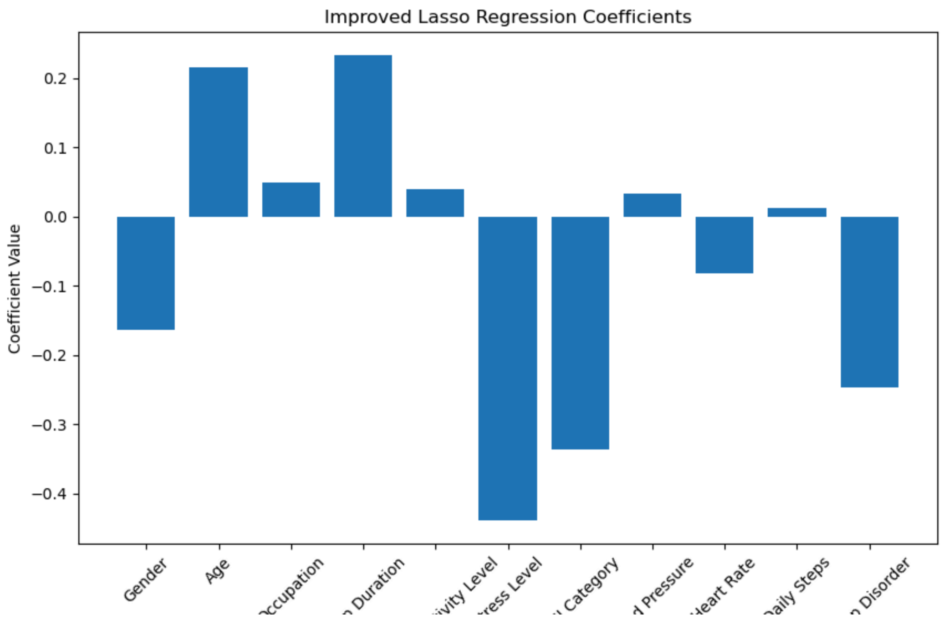
However, in the first run, we observed a lower-than-expected R^2 score, which suggested that the default regularization strength (alpha) might have been too restrictive. To alleviate this, in the second run, we used GridSearchCV to perform hyperparameter tuning and automatically search for the optimal alpha value. This adjustment allowed the model to retain informative features by reducing unnecessary penalization. The effect of this change was visible not only in the improved R^2 score but also in the spread and values of the resulting feature coefficients.

When comparing Lasso to other algorithms in our study—such as Linear Regression and Random Forest—it becomes clear that Lasso prioritizes simplicity and interpretability over raw predictive power. While Random Forest achieved a higher R^2, it did so with a complex ensemble structure, whereas Lasso gave us direct insight into the importance of each feature. This trade-off makes Lasso valuable in the scenario where understanding the model's decisions is just as important as prediction accuracy.

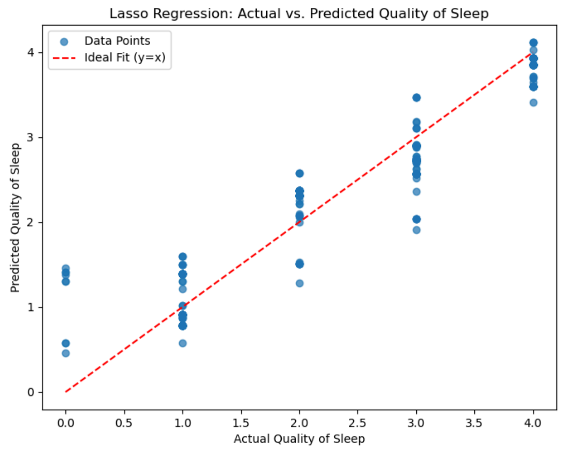
1st Lasso Run Coefficients:

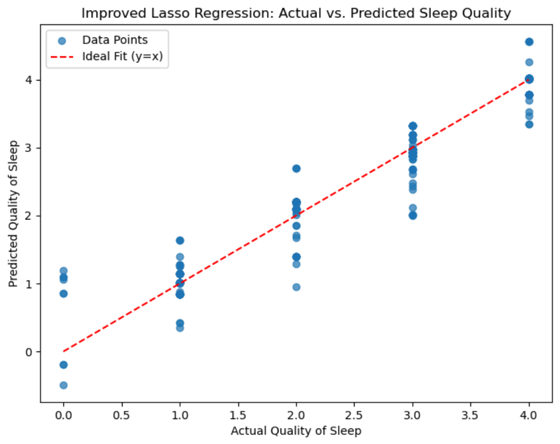


2nd Lasso Run Coefficients (with GridSearchCV)



It is worth noting that dimensionality reduction and feature selection does not always work best, especially in an already very-well preprocessed dataset environment (such as this one).





Conclusion

In summary, we selected a sleep study dataset and chose the sleep quality as our predictive target. Then, we performed 5 regression algorithms over the dataset, over which we did 2 runs for SVR and Lasso regression. The results of these algorithms, in terms of R2 score, are surprisingly very high (almost all >= 0.9), which indicates that the models are performing reasonably well. The Random Forest achieved the highest R2 score, closely followed by linear and Support Vector Regression(SVR). Decision tree had a high accuracy but also a high chance of overfitting, especially for smaller datasets. The Lasso regression lags slightly behind on performance, but it offers insights into attributes and target. Overall, this project demonstrates the different models of regression and their performance, and also highlights the different aspects of their predictive power.

Reference

Sleep Health and Lifestyle Dataset

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