Intelligent Systems - Sleep Recommender

Kristian van Kuijk i6214757, Pavan Aakash Rangaraj i6214168, Boris Borisov i6211839

I. Introduction

Sleep plays an important role in your physical health. This paper states how Recommender for Sleep was implemented, while taking into account our course in Intelligent Systems. A lot of students report daytime sleepiness and around 70 percents attain insufficient sleep. The lack of sleep can result in lower grades on the exam, increased risk of academic learning, compromised learning, increased headache and impaired mood as stated by Ronald D et al. [7] Our aim is to boost the student community' lifestyle and sleep quality through a system that would tell you what was your sleep score, and explain how to improve it. Our expert systems helps improve decision of the user on his lifestyle. We combine data and knowledge with reasoning. Besides, a machine learning model and a genetic algorithm were implemented, which we will explain in the later section.

II. User Interface

The graphical user interface was implemented in Python. The Python standard GUI package Tkinter was used [1]. The Interface required the following inputs from the user: the age, what time the user went to sleep, what time the user woke up, the number of steps the user completed during the day and the number of caffeinated drinks the user had in the day (see figure 1).

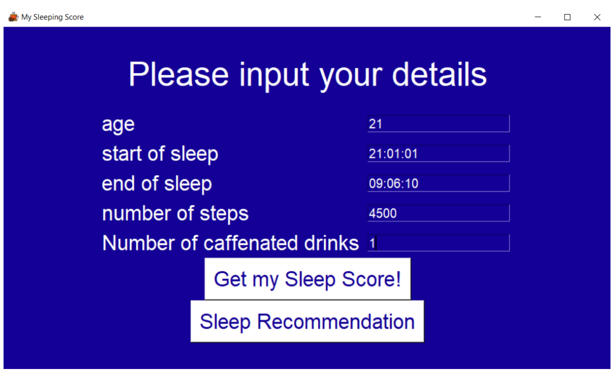

Fig. 1. The user can provide the details here

After the user provides these details, the two following options are available:

- a sleep quality score

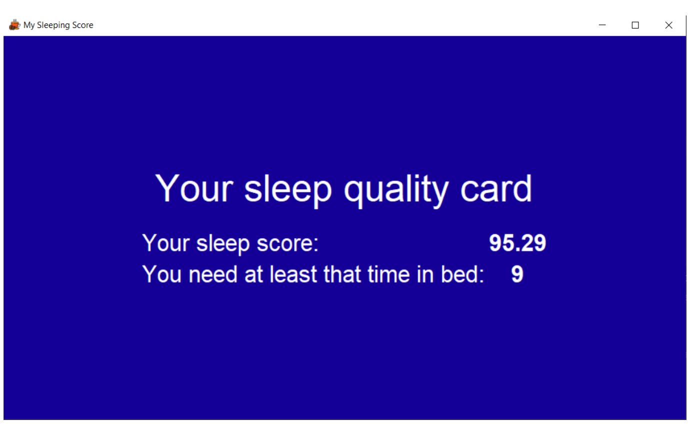

Fig. 2. Sleep quality score of the user

- a recommendation on how to improve his sleep score if it was too low

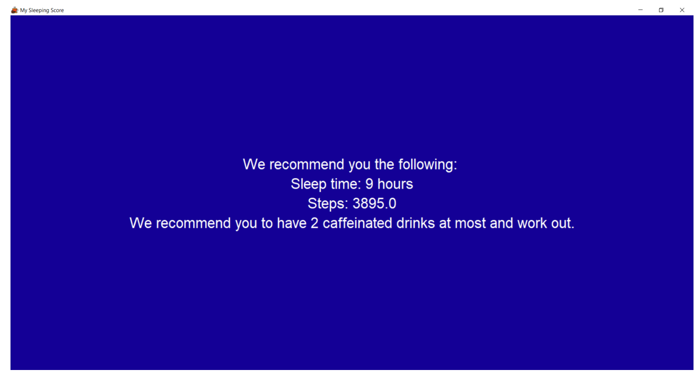

Fig. 3. Recommendation for the user based on the data from the previous day

Th number of hours of sleep we recommend to the user is rule based from knowledge we have. The knowledge was acquired form a paper by Jean-Philippe Chaput et al. [3]

III. Machine Learning

The most used sleep trackers use a mathematical formula to compute a sleep score. The *Fitbit* sleep score for instance is computed by the sum of your individual scores in sleep duration, sleep quality, and restoration. This was not convenient for our system as we would need to compute individual scores for each of those categories. Instead, we built a random forest model to predict the sleep score. Random forest models are known for many advantages (usually reduces overfitting and are robust to outliers for instance). The data was acquired between 2014-2018 (so quite recent) and collected through the cycle app on iOS. The dataset is made of approximately 700 entries, which means the dataset is relatively small. After features selection (the correlation matrix is shown in figure 4) and considering some features can more easily be asked to the user (it might make more sense to ask the time the user went to bed instead of his heart rate). The following features were used to train the model on: *Time in bed* (how long did the user sleep), *Start time* (what time did the user fell asleep), *End time* (what time did the user wake up), *Activity* (steps), *External factor* (number of caffeinated drinks) based on how easily it could be provided by the user, and the influence of such a feature in predicting the Sleep Score. Lastly, hyperparameter tuning was performed using a randomized search, with 10 folds for each of 100 candidates, totalling 1000 model fitting. This means not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The main advantages of such a method over a grid search is that the algorithm searches over distributions of parameter values rather than a predetermined lists of candidate values for each hyperparameter. Being able to search over hyperparameter distributions also allows you to be more opinionated about what you expect a hyperparameter's best value to be by specifying a distribution (for instance normal, uniform, poisson...). The parameters and their distribution can be found in table I. The best model (with a maximum depth of 3, minimum samples leaf of 3, minimum samples split of 10 and number of estimators of 269) had a root mean square error training score of 9.47 and a root mean square test score of 10.67. This is still high knowing the sleep score ranges from 0 to 100. The detailed parameters and score can be found on figure 5.

Feature	Distribution Sampled
Number of estimators	Uniform between 50 and 500
Maximum depth	Uniform between 3 and 10
Minimum samples split	Uniform between 2 and 20
Minimum samples leaf	Uniform between 1 and 10

TABLE I
FEATURES AND THEIR DISTRIBUTION

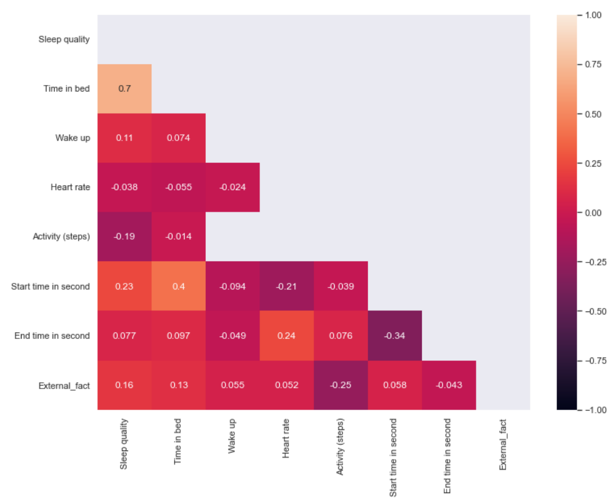

Fig. 4. Correlation Matrix

Fig. 5. Hyperparameter Tuning Including Maximum Depth, Minimum Samples Split, Minimum Samples Leaf and Number of Estimators (scoring is the negative root mean square error, hence higher the better)

IV. GENETIC ALGORITHM

An expert system must be able to not only make a decision using the information provided by the user but also be proactive and help users take the right decision providing additional information like a decision support system. Therefore, in addition to our Random forest model that estimates the users sleep score given the features mentioned in Fig: 1, we included a Genetic Algorithm that was designed to take in as input the users previous night's sleep score, using which we will suggest practices that lead to higher sleep scores. A genetic algorithm was chosen for this task based on the Building Block Theorem explained by Salomon [6].

The genetic algorithm takes in as input the users previous sleep score and sets a lower-bound on the fitness of the initial population as specified in Table II. This strategy forces the GA to converge to higher quality results than it would have normally done if the initial population was generated randomly.

Previous Night Sleep Score	Initial Population Fitness Lower-bound
score $\leqslant$ 30	77
score > 30	60

TABLE II
LOWER-BOUND ON INITIAL POPULATION FITNESS

Once the lower-bound is set, the initial population is generated. The initial population is encoded as a 5-variable array. It consists of the same 5 variables the Random forest model uses as input, namely, **Time in bed** (which ranges from 5-12 hours based on [4], **Sleep start time,Wake up time** (start time and wake up time range over 24hrs), **Number of steps** (ranges from 2000-7500 steps as recommended by Driver et al. [2] and **Number of caffeinated drinks** (which range from 0-5 as explained by Snel et al. [8].The encoding although simple and direct had some limitations in the crossover strategies that could be applied on the chromosome which we will see in a later part of this report.

The size of the initial population and the number of generations were set after some trial and error experiments taking into consideration run-time and optimal results. The population size was set to 100 and the number of generations was set to 10. The results of the experiments can be found in Tabel:III and Figure 6.

The fitness of the population can be conveniently measured by the Random forest model, which will provide a score between 0 and 100, which is equivalent to the percentage of success for that particular individual. Thus, we conclude that this is a very effective fitness measure.

During the crossover process it is very important to deal with destruction. That is why we copy the 50 best individuals onto the next generation to avoid the loss of good solutions. The remaining 50 spots of the new generations will comprise of children created by crossing over individuals selected with replacement through Roulette Wheel selection from the previous generation. The probability of each individual being selected was proportional to their fitness.*"The basic advantage of proportional roulette wheel selection is that it discards none of the individuals in the population and gives a chance to all of them to be selected. Therefore, diversity in the population is preserved. However, proportional roulette wheel selection has few major deficiencies. Outstanding individuals will introduce a bias in the beginning of the search that may cause a premature convergence and a loss of diversity. For example, if an initial population contains one or two very fit but not the best individuals and the rest of the population are not good, then these fit individuals will quickly dominate the whole population and prevent the population from exploring other potentially better individuals. Such a strong domination causes a very high loss of genetic diversity which is definitely not advantageous for the optimization process" Razali et al [5].* We will see how this strategy affects our output in the results sections.

Once the individuals were selected for crossover, the resulting child would have (*parent1 value* + *parent2 value*) / 2. Due to the nature of our encoding, where each feature had different ranges that were legal, crossover strategies such as Uniform crossover or Headless Chicken crossover could not be applied.

The newly generated chromosomes are subjected to a 5% mutation rate. This rate is higher than the usual 1-2% to compensate for the possibility of premature convergence from the roulette selection strategy. A higher mutation rate might help restore diversity in such cases.

A. Experiments and Results

To determine the number of initial population and number of generations we run a trial and error experiment. We can see that the best chromosome is generated at around 10 generations for a population size of 100 (Figure:6). This could be an indication of premature convergence, but since the fitness score for that chromosome was 97.86, which is pretty high given that the fitness range is from 0-100 we decide not to make changes to the model.

Number of Generations	Run-time in seconds
25 Generations	20.64s
50 Generations	58.7s
100 Generations	100.6s

TABLE III
NO OF GENERATIONS VS RUN-TIME

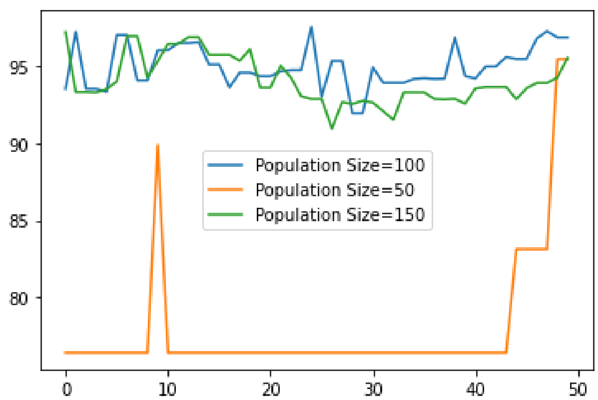

Fig. 6. Evolution of best chromosome over different initial population sizes

Considering the behaviour seen in Table III and Figure:6 the final model of the Genetic Algorithm consists of the following parameters: simulation of 100 chromosomes over 10 generations with a mutation rate of 5%.

V. CONCLUSION

This report presented our expert system that was implemented. It helps improve the lifestyle of a person by combining data and knowledge with reasoning.
To compute the sleep score, a random forest regressor model was fitted on data from the cycle app on iOS. After features selection and hyperparameter tuning, the best model got a root mean square error of 10.67 on the test set (considering the sleep score ranges from 0 to 100).
Besides, the user can then use our GA to get daily recommendations in the form of number of steps, number of caffeinated drinks, recommended hours of sleep.

REFERENCES

[1] Graphical user interfaces with tkinter in python.

[2] Helen S Driver and Sheila R Taylor. Exercise and sleep. *Sleep medicine reviews*, 4(4):387–402, 2000.

[3] Caroline Dutil Jean-Philippe Chaput and Hugues Sampasa-Kanyinga. Sleeping hours: what is the ideal number and how does age impact this?, Nov 2018.

[4] Consensus Conference Panel, Nathaniel F Watson, M Safwan Badr, Gregory Belenky, Donald L Bliwise, Orfeu M Buxton, Daniel Buysse, David F Dinges, James Gangwisch, Michael A Grandner, et al. Recommended amount of sleep for a healthy adult: a joint consensus statement of the american academy of sleep medicine and sleep research society. *Journal of Clinical Sleep Medicine*, 11(6):591–592, 2015.

[5] Noraini Mohd Razali, John Geraghty, et al. Genetic algorithm performance with different selection strategies in solving tsp. In *Proceedings of the world congress on engineering*, volume 2, pages 1–6. International Association of Engineers Hong Kong, China, 2011.

[6] Ralf Salomon. Short notes on the schema theorem and the building block hypothesis in genetic algorithms.

[7] Ronald D Chervin Shelley D Hershner. Causes and consequences of sleepiness among college students. 6(7):73+, jun 2014.

[8] Jan Snel and Monicque M Lorist. Effects of caffeine on sleep and cognition. *Progress in brain research*, 190:105–117, 2011.