Capstone Project
Machine Learning Engineer Nanodegree

Shohei Nagamine
March 15, 2018

1 Defenition

1.1 Project Overview

The day after I couldn't sleep well makes me unhappy. It makes me unrefreshed, tired, forgettable, slow, short-tempered and so on. Several studies show the importance of getting adequate sleep. According to an article of "Sleep, Learning, and Memory" in Healthy Sleep, a research suggests sleep itself has a role in the consolidation of memory, which is essential for learning new information (*1). Some studies reveal that sleeping five hours or less per night increased mortality risk from all causes by roughly 15 percent (*2). Moreover, according to an article of CBS NEWS, drivers who get just one to two hours less than the recommended seven hours of sleep in a 24-hour period nearly double their risk for a car crash (*3).

Therefore, taking time for sleep is obviously important. However, although we spend enough time for sleep, sometimes we cannot sleep well and cannot get the advantages of it. According to the National Institute of Neurological Disorders and Stroke, about 40 million people in the United States suffer from chronic long-term sleep disorders each year and an additional 20 million people experience occasional sleep problems (*4). Therefore, it seems to be valuable to know what brings us good sleep. That way, a lot of people can live healthier high-quality life without wasting our time for not efficient sleep.

In this project, I achieve that mainly by making a valid model which predicts people's sleep quality by using the dataset of "2013 SLEEP IN AMERICA POLL" (*5) provided by national sleep foundation and finding features which are important to predict sleep quality.

1.2 Problem Statement

To know what brings us good sleep, I'd like to find out important features to predict self-reported sleep quality by making a valid machine learning model from the dataset of "2013SleepinAmericaPollExerciseandSleepRawDataExcel" (* 5) provided by National Sleep Foundation. Then, I will find out whether the important features are positively or negatively correlated with sleep quality. That way, we can know what we are recommended to do and not recommend to do to make their sleep quality better..

Since the target variables are discrete and have multiple classes, this machine learning problem is a multiclass classification problem, because "**q30**" is a target variable and represents a respondent's sleep quality by multiple classes, and this value will be predicted by several input variables such as the ones represent the length of sleep time, whether a respondent drinks alcoholic or caffeinated beverages, the frequency of doing exercise and so on.

1.3 Metrics

In short, I chose F1_score with macro-average as a metric for this problem. Here is a reason for that.

This is a multi classification problem and this doesn't attach importance to precision nor recall, so using accuracy as a metric was one of the candidates I came up with. However, since the values of the target variable are highly skewed and not uniformly distributed, using accuracy as a metric is not appropriate. As **Table 1** suggests, more than half of values are skewed to "2" in **q30**, so if I make a model which always predicts "2", the accuracy will be 0.569. Since this is a bad model, the evaluation of the model should be much lower.

Table 1 – Distribution of values of the target variable which represents sleep quality **(q30)**.

Description of class	Class of target variable	Each number of lines
Very good	1	190
Fairly good	2	569
Fairly bad, or	3	192
Very bad	4	49

Thus, we need a metric which evaluates a model lowly in this case, and I think F1 score with macro-average will do the job. F1 score with macro-average is a metric which doesn't take label imbalance into account and lowers its score if either precision or recall is bad. I tested a model which always predicts "2" with the metric, and it scored 0.1813. Since the model is obviously bad, the score of 0.1813 seems to be valid and acceptable. I also tried F1 score with micro-average which takes label imbalance into account. It scored 0.569 which is exactly same as the accuracy of the same model, so I think this is not a suitable metric for this problem.

Table 2 - Scores of the model which always predicts "2" for **q30** on each metric. (To obtain the result, execute "experiment_metrics.py")

Metric	Score
Accuracy	0.5690
Precision (micro average)	0.5690
Recall (micro average)	0.5690
F1 Score(micro average)	0.5690
Precision (macro average)	0.1422
Recall (macro average)	0.2500
F1 Score(macro average)	0.1813

From now on, F1 score in this report means F1 score with macro average.

2 Analysis

2.1 Data Exploration

About the raw dataset

The dataset I will use for making the model is "2013SleepinAmericaPollExerciseandSleepRawDataExcel" from National Sleep Foundation (* 5). According to "Summary of findings"(* 6) of this study, this dataset is made by National Sleep Foundation and WB&A Market by conducting a national survey of Americans regarding their sleep habits. This poll is an annual review of habits, behaviors and attitudes pertaining to sleep and sleep quality. The study includes measures of sleepiness, drowsy driving, sleep disorders and general health.

The most of the dataset is made by answering the questions of "SIAQuestionnaire2013.pdf" (* 7). For example, the column of **q1** in the dataset contains the answers of the question 1 in the questionnaire. If the answer to the question is "01", "1" will be in the cell. However, some answers for the questionnaire are converted to continuous values such as **Q1VALUE**, **Q2VALUE** and so on. Since it's impossible to detail all the features in a permissible length, please refer the "SIAQuestionnaire2013.pdf" (* 7) for the details of the features. However, I will introduce some of the features which I guess relevant to predict the respondents' sleep quality.

Table 3– Some important features which I guess relevant.

Column name	**Type**	**Description**
q29c	continuous	Quantity of drinking caffeinated beverages between 5:00 PM and 5:00 AM the next morning in 12 ounce servings.
Q38	continuous	Amount of time a respondent spent for moderate physical activities per day in the past 7 days in minutes.
q20	classification	If a respondent snores loudly or not.

Basically, the features begin from "**q**" contain classification value though **q29c** in **Table 3** is one of the exceptional cases, and the ones begin from "**Q**" contain continuous values.

<u>About input variables</u>

I will remove following unnecessary features from the input variables.

- features which obviously not correlated with target variables.(Ex. **caseid**, **source**, **market**)
- Duplicate features.
- Features which are not in use.
- Features which I cannot understand what they represent.
- Features which are not important to predict target variables (Feature Selection), because the one of machine learning algorithms I use is sensitive to noise and they will lower the performance.

Furthermore, I will create some additional features because I guess they are relevant to predict sleep quality.

Table 4 – The features I will create from the raw dataset.

Feature name	**Type**	**Description**	**Calculation**
Q43TOTAL	Continuous	Minutes per day the respondents spend sitting during activities in the past 7days	Sum of **Q43A**,**Q43B**,**Q43C**,**Q43D**,**Q43E**,**Q43F**,**Q43G1**,**Q43G2**,**Q43G3**
Q36Q38Q40TOTAL	Continuous	Total minutes the respondents spent for all intensities (light,moderate,vigorous) of physical activities per day in the past 7 days.	Sum of **Q36**,**Q38**,**Q40.**
Q3Q1DIF	Continuous	Time difference of going to bed between nights before workdays or weekdays and the ones the respondents do not work the next day or weekends.	Absolute difference of **Q3VALUE** and **Q1VALUE**
Q4Q2DIF	Continuous	Time difference of getting up and out of bed for good before workdays or weekdays and the ones the respondents do not work the next day or	Absolute difference of **Q4VALUE** and **Q2VALUE**.

		weekends	
Q4Q2DIFQ3Q1DIFTOTAL	Continuous	Sleep time difference between nights before workdays or weekdays and the ones the respondents do not work the next day or weekends.	Sum of **Q4Q2DIF** and **Q3Q1DIF**.

<u>**Characteristic about the dataset**</u>

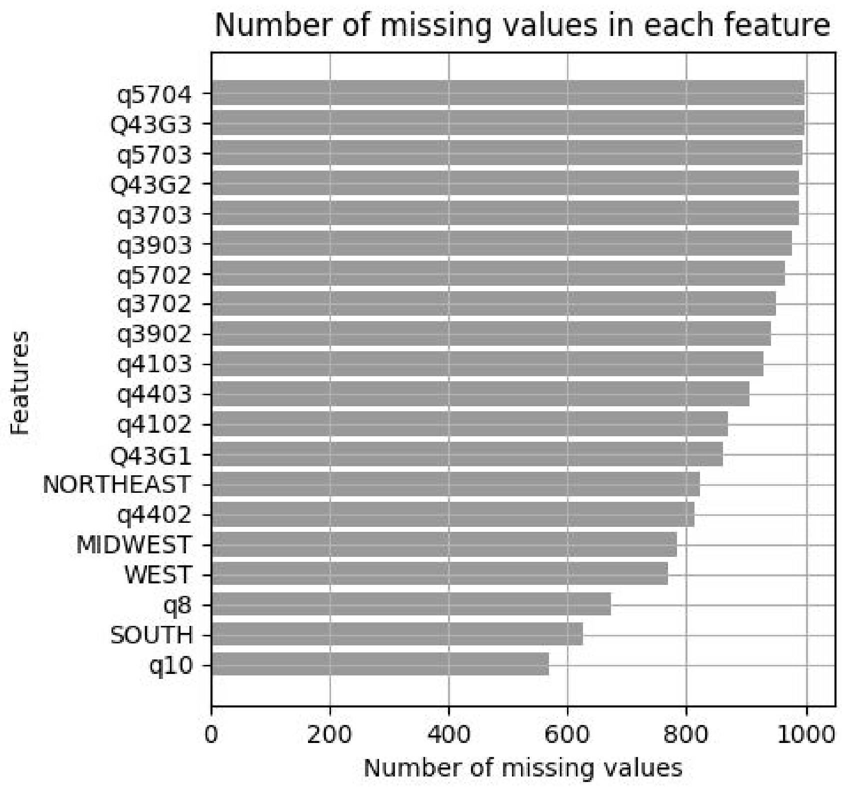
Here are some statistics about the entire dataset.

Table 5 – Some statistics about the dataset and features I will use.

The number of instances.	1000
The number of features in the raw dataset	256
The number of features I will use including a target variable in the raw dataset	124

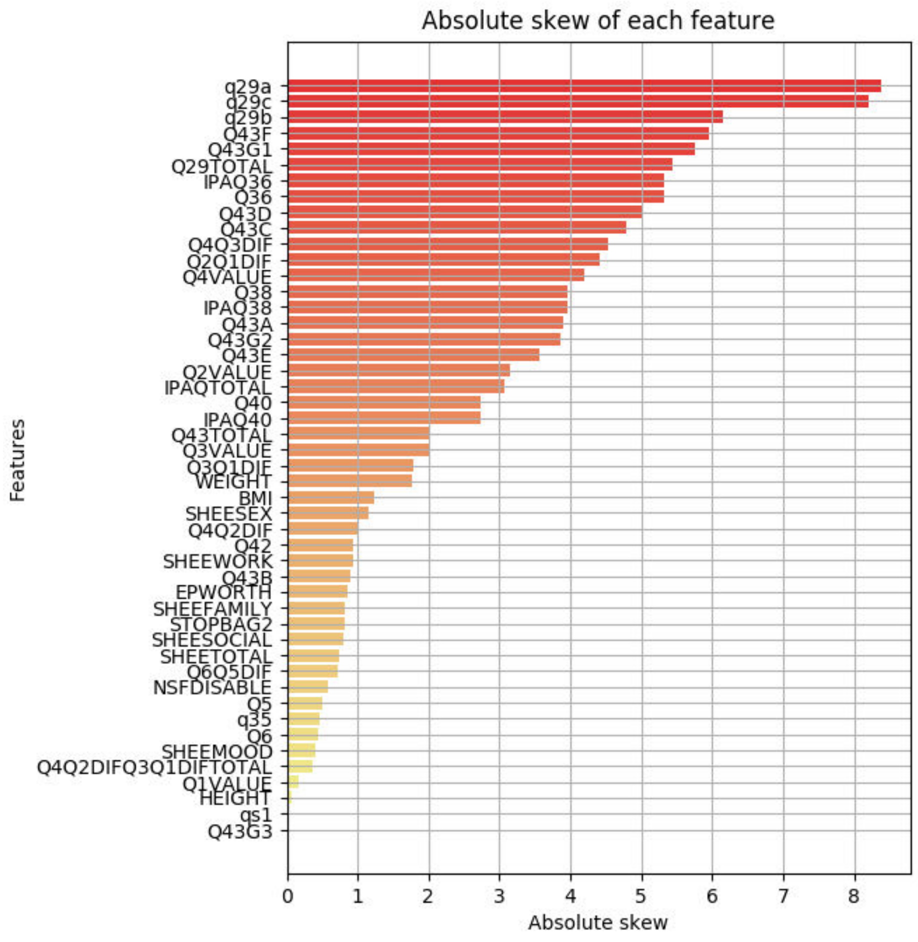
There are a lot of missing values. **Fig 1** shows input variables which have missing values more than 500.

Fig 1 - The input variables which have missing values more than 500.



Some features which contain continuous values are highly skewed (**Fig 2**).

Fig 2 – Absolute skew of each feature mainly calculated by scipy.stats.skew (*8)



Also, some data have outliers. For example, the maximum value in the feature of **Q36** is 1440, but as you see the **Fig 3**, the maximum value is obviously an outlier and the data is highly skewed toward the range between 0 and 200. I can see this kind of distribution in other features such as **q29a (Fig 4)** and **Q43F (Fig 5)**

Fig 3 – **Q36**: Minutes the respondents spend for vigorous exercise per day in the past 7 days

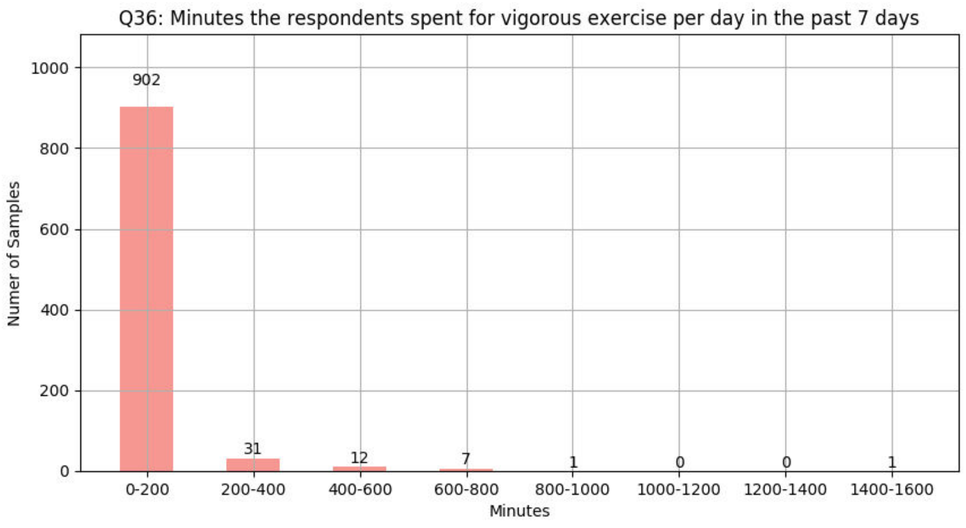


Fig 4 - **q29a**: Number of 12 ounce servings of caffeinated beverages the respondents take between 5:00 AM and noon on an average weekday or workday. The maximum values in **q29a** is 60, however, 60 is obviously an outlier.

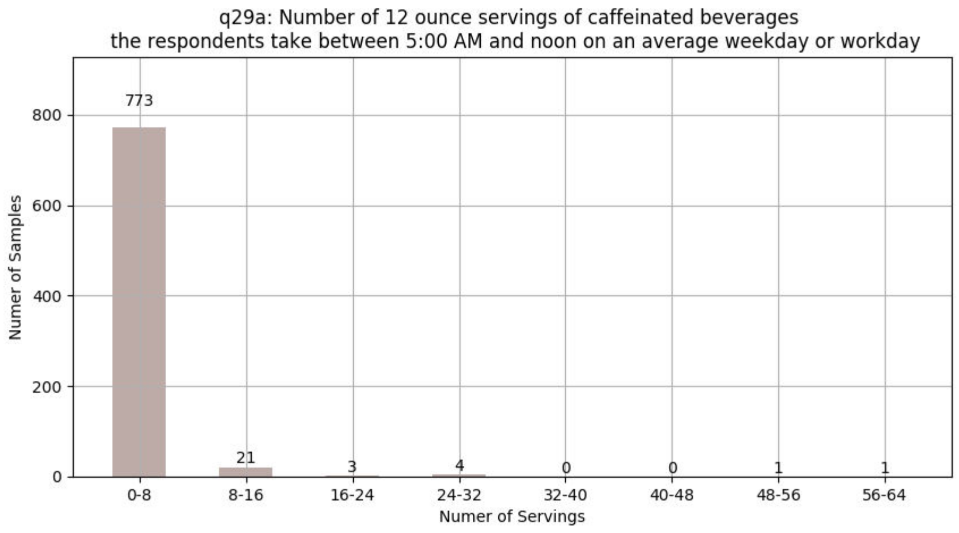
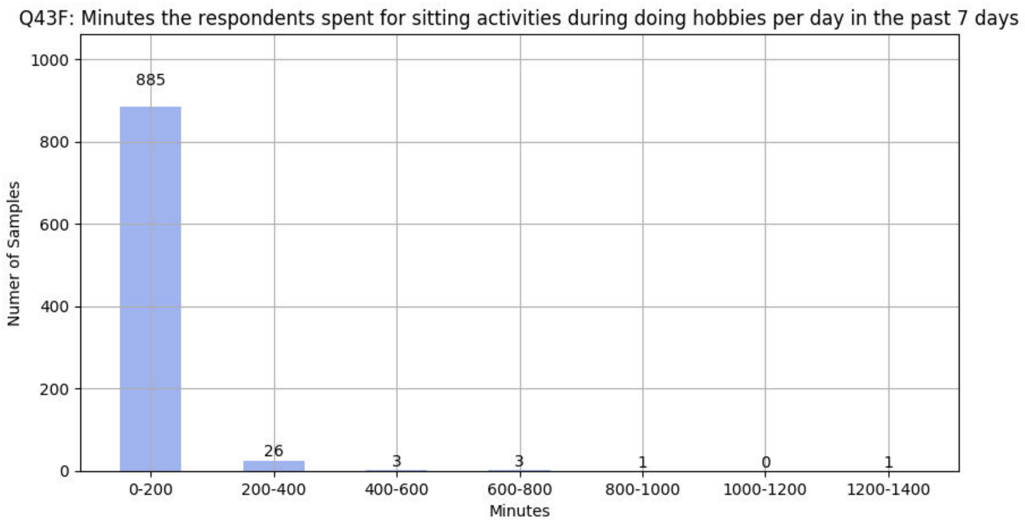


Fig 5 - **Q43F**: Minutes the respondents spent for sitting activities during doing hobbies per day in the past 7 days. The maximum values in **Q43F** is 1200, however, 1200 is obviously an outlier.



For target variable, since my aim is to find out some features which are relevant to sleep quality, I think following feature is appropriate.

Table 6 – About a target variable

Name of a feature	**Type**	**Description**
q30	Classification	The overall sleep quality during the past two weeks

However, I will convert the values of target variables as **Table 7** to make the analysis easier. Since there are no values of 98 or 99, I don't (can't) apply any conversions for those values.

Table 7 – Value conversion of the target variable

Answer	Convert from	Convert from
Very good	1	4
Fairly good	2	3
Fairly bad, or	3	2
Very bad	4	1

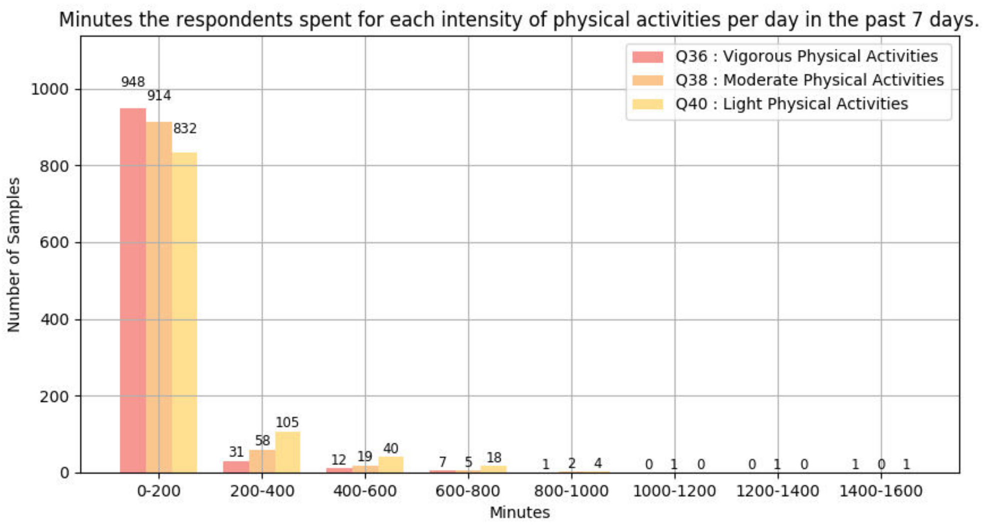
<u>2.2 Exploratory Visualization</u>

In this section, I will enumerate the features which I guess relevant to predict the respondents' sleep quality. Since this project aims to give some hints for people who cannot sleep well although they take enough sleep time, I won't enumerate the features which represent sleep hours.

- Q36, Q38, Q40 (Amount of various intensities of physical activities per day in the past two weeks):

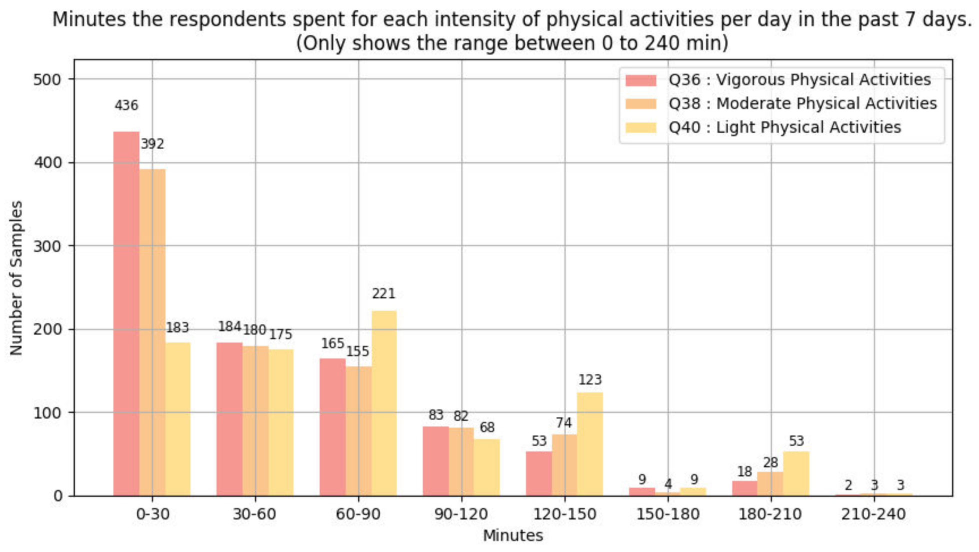
Q36, Q38,Q40 respectively indicate the amount of vigorous, moderate, light physical activities per day in the past two weeks. I guess everyone might have experienced when we physically get tired, we often can sleep deeper. A research also suggests it. According to the research by the team in Department of Neurology in Northwestern University, aerobic exercise improves self-reported sleep and quality of life in older adults with insomnia (*9). For these reasons, I guess amount of physical activities is relevant to predict sleep quality..

Fig 6. Q36, Q38,Q40 - Minutes the respondents spend for each intensity of physical activities per day in the past 7 days.



As **Fig 6** in the section of Data Exploration shows, the features are highly skewed and the most of the samples are in the range of 0 to 200 minutes, so I also provide a figure that only shows the range between 0 to 240 (**Fig 7**).

Fig 7. Q36, **Q38**,**Q40** – Minutes the respondents spend for each intensity of physical activities in the past 7 days. It only shows the number of samples in the range between 0 to 240 min.



About 40 % of people don't spend their time for moderate and vigorous physical activities more than 30 minutes per day. On the other hand, the ratio of people who spend their time for light physical activities less than 30 minutes is only 18.3 %. Moreover, 64.9% of the people spend their time for light physical activities more than 30 minutes. From the visualization, I can see people tend to do light physical activities more than the ones vigorous or moderate.

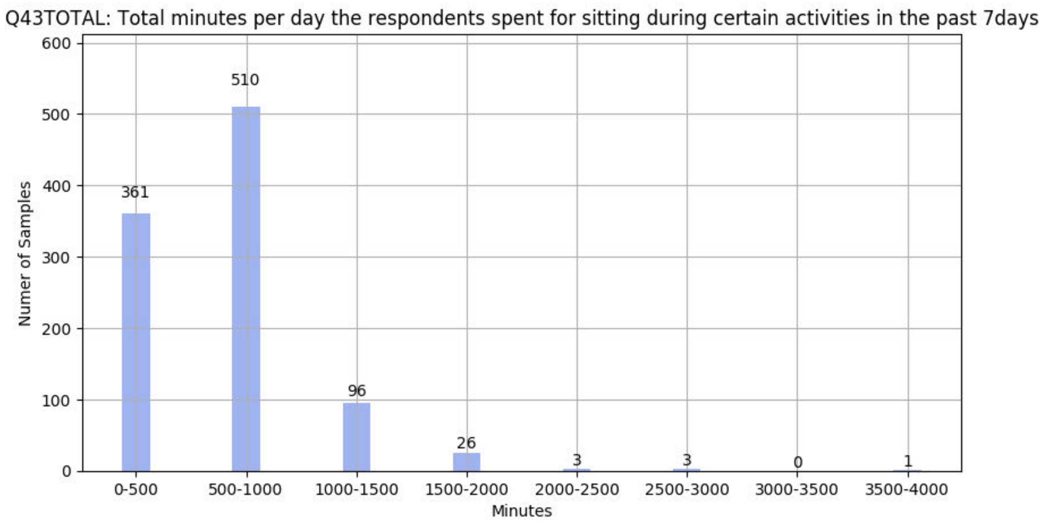
-Q43TOTAL (Minutes a respondent spend for sitting per day in the past 7 days):
If physical activities improve sleep quality, I guess minutes a respondent spend for sitting is also relevant to predict the respondent's sleep quality, because if people spend their time for sitting for many hours, I guess it tends to lead to a lack of physical activities. For that reason, I made a new feature called **Q43TOTAL** which is sum of **sitting** minutes per day that each respondent spend for in the past 7 days. In other words, this feature is sum of following sitting activities (features):

--**Q43A**: Watching television,
--**Q43B**: Using a computer,

--Q43C: Reading,
--Q43D: Socializing with friends or family,
--Q43E: Traveling in motor vehicle or on public transport,
--Q43F: Doing hobbies,
--Q43G1~ G3: Something else.

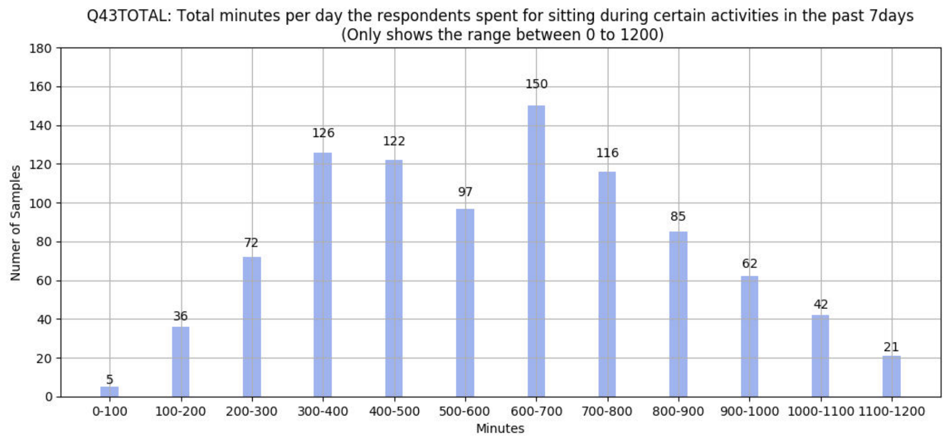
For the answers of "Refused" or "Don't know" in each feature, I imputed the mean value of each feature.

Fig 8. Q43TOTAL - Total minutes per day the respondents spent for sitting during certain activities in the past 7days



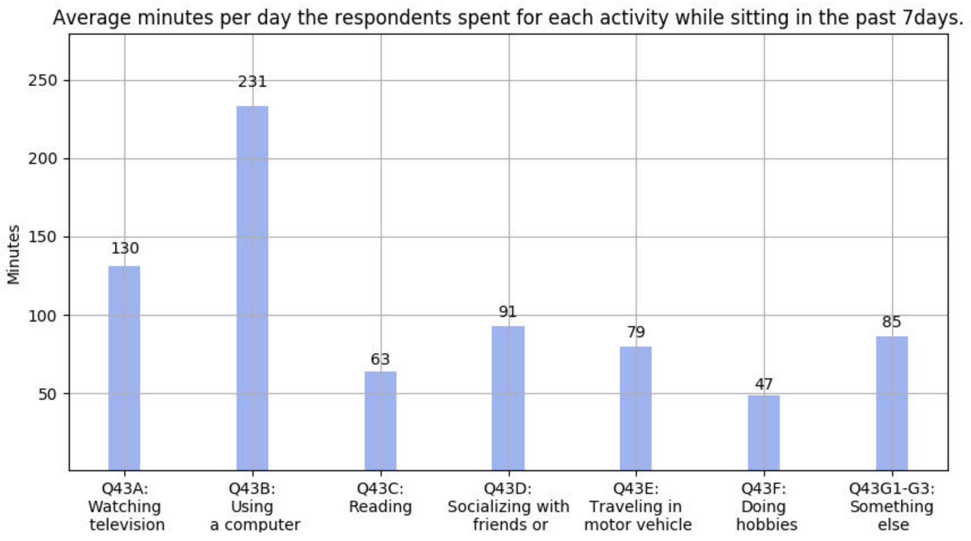
As **Fig 8** suggests , the most of samples are in the range between 0 to 500 and 500 to 1000, so to show the detail of the samples, I made a figure that only shows the range between 0 to 1200.

Fig 9. Q43TOTAL - Total minutes per day the respondents spent for sitting during certain activities in the past 7 days. It only shows the samples in the range between 0 to 1200 min.



The most of samples belong to the range between 200 to 900 minutes, and the range between 500 to 600 minutes which is about 8 hours 20 minutes to 10 hours has the largest samples. That's very long time and seems to be unhealthy. What does make people sit for such a long time? **Fig 10** reveals that.

Fig 10. Average minutes per day the respondents spent for each activity while sitting in the past 7days.



As you might guess, people spend their sitting time for using computers for over 3 to 4 hours on average. It makes sense because nowadays, we do a lot of activities on PC.

- q29c (Amount off caffeinated beverages a respondent took between 5:00 PM and 5:00 AM the next morning)

According to an article titled "How long are you affected by caffeine?" from ScienceNordic (*10), Olav Spigset says the time it takes for the caffeine level to drop by half is an average of five hours. After 10 to 15 hours, there is no longer enough left to have an impact. If so, drinking caffeinated beverages in early in the day doesn't affect much for sleep quality, but drinking them within 10 hours before going to bed negatively may affect sleep quality especially within 5 hours before that. According to q1 and q3 in the dataset, we can see the most of people get into bed at night. (7:00PM~5:00AM)

Fig 11. q1 – Time the respondents get into bed on nights before workdays or weekdays

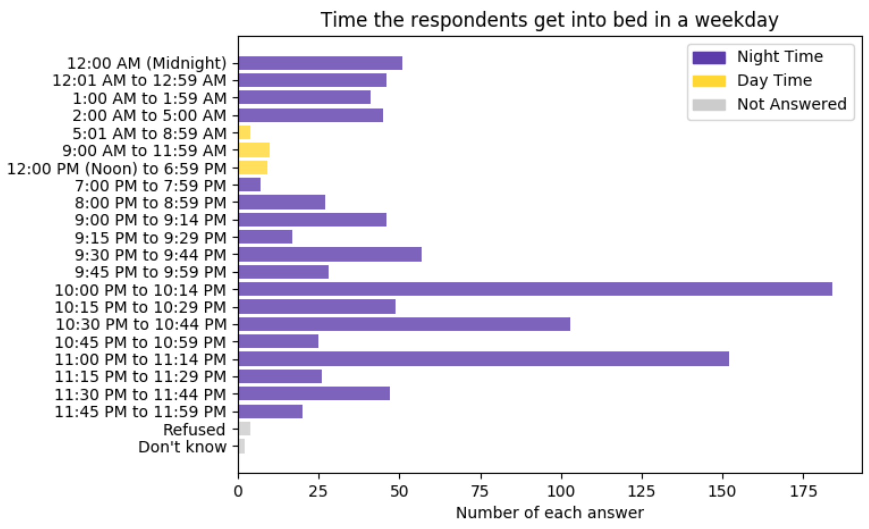
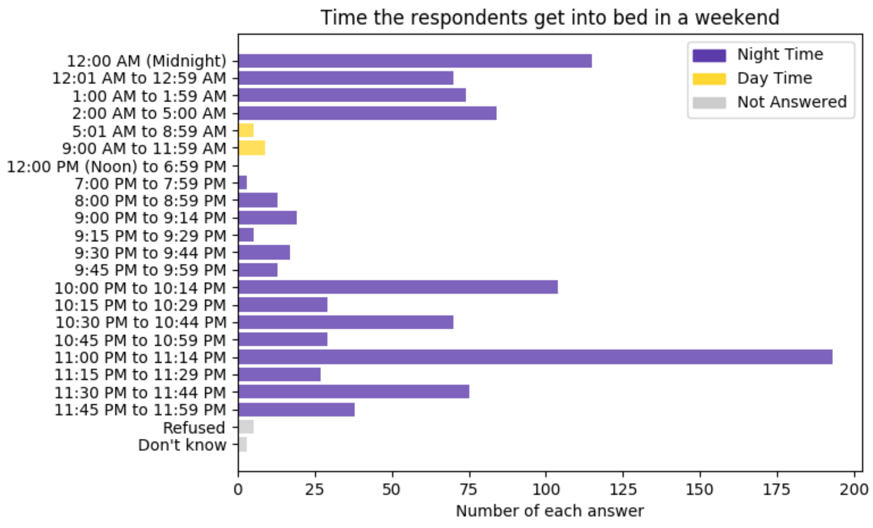
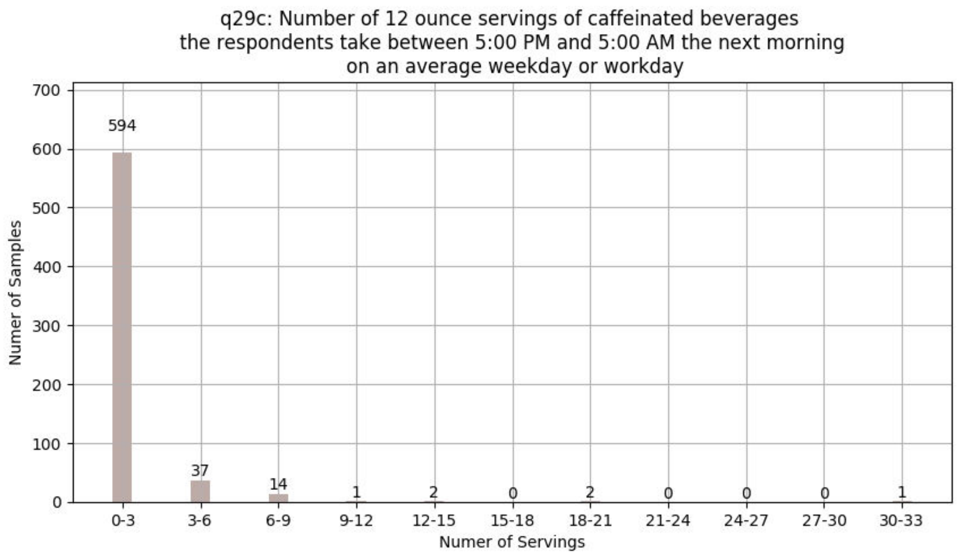


Fig 12 q3 - Time the respondents get into bed on nights you do not work the next day or weekends



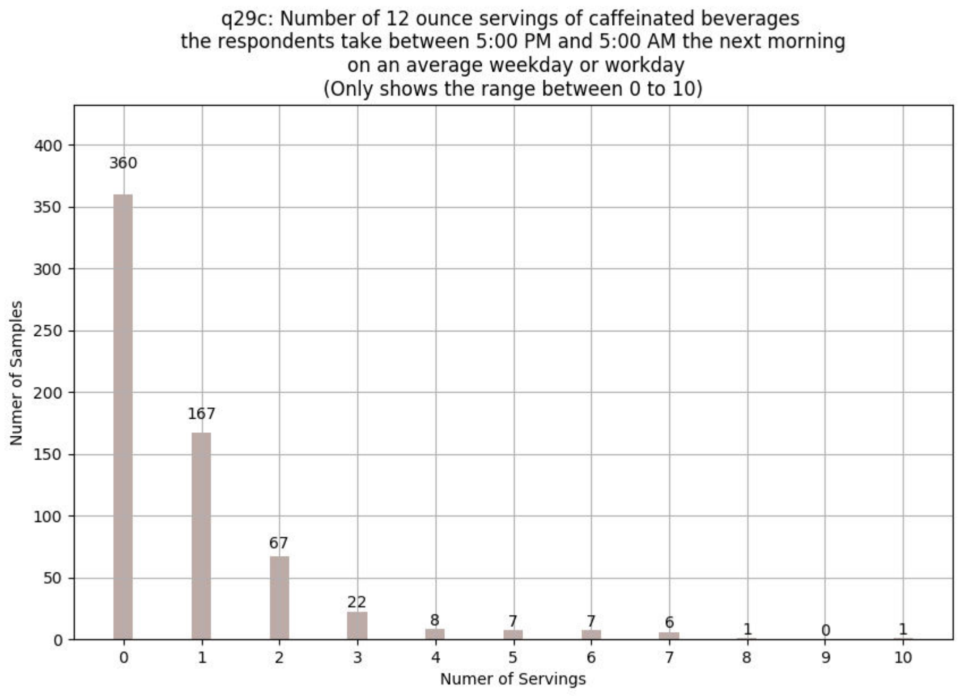
If people drink caffeinated beverages during this period of time or the time close to the period of time, I guess it lowers their sleep quality, because there is high possibility that caffeine they took still remains in their bodies and has the negative effect even when they get into beds.

Fig 13. q29c - Number of 12 ounce servings of caffeinated beverages the respondents take between 5:00 PM and 5:00 AM the next morning on an average weekday or workday



The most of the respondents drinks 0 to 3 servings between 5:00PM and 5:00AM, so let's focus on this range more.

Fig 14. q29c - Number of 12 ounce servings of caffeinated beverages the respondents take between 5:00 PM and 5:00 AM the next morning on an average weekday or workday. It only shows the number of samples in the range between 0 to 10 servings.



36 % of people don’t drink caffeinated beverages during this period of time. 25.6 % of people drink 1 to 3 servings of caffeinated beverages.

2.3 Algorithms and Techniques

I will use the algorithms which meet following conditions as much as possible.

1, Able to handle a lot of input variables.
The number of input variables I use is 123, so the algorithms need to perform well with a lot of input variables.

2. Gives estimates of what variables are important in the classification.
Since my aim is to find out the features which are capable of improving our sleep quality, the ability to show important features to predict target variables is necessary,

3、Less sensitive to outliers-
As I mentioned in **Data Exploration**, the data seem to contain a lot of outliers. I can reduce the effect of outliers by using log transformation or simply removing the outliers by Local Outlier Factor, still the algorithms which are robust to outliers are preferable.

4, Robust to noise
Since the project aims to find out the key features which improve sleep quality from a lot of features which I don't know if they are meaningful or less meaningful, so there may be a lot of noise. I can reduce the noise by selecting important features, but the algorithms which are robust to noise are preferable.

5、Robust to missing data
As I mentioned in **Data Exploration**, the input variables contains a lot of missing values, so the algorithms which perform well with a lot of missing values are preferable though I will apply imputation for missing values in pre process.

I think the algorithm which meets all these conditions is Random Forests. Random Forests is an ensemble classifier that consists of multiple decision trees trained by a dataset and outputs the class that is the mode of the classes predicted by each Decision Tree.(*11) I will use the one in sklearn (*12). I will use the default parameters except for n_estimators and random_state. I will find the best value for n_estimators by using GridSearchCV (*13) and also input various values to the parameter of random_state to make the results stable.

AdaBoost with Random Forests (AdaBoosted Random Forests) is also candidate. AdaBoost is a kind of ensemble technique. First, AdaBoost classifies samples by using the base model which requires predicting target variables better than random guessing. In AdaBoosted Random Forests, the base classifier is the one made by Random Forests. Then, there will be two kinds of samples which the base model classifies correctly and incorrectly. Adaboost focuses on the samples which are classified incorrectly and remakes the model to classify the misclassified samples correctly. Adaboost continues the process of the classification and the remake over and over until we get a good sufficient model.

According to the page of AdaBoost in NickGillianWiki (*14) , Adaboost can be sensitive to outliers and noise and it doesn't meet the condition 3 and 4. However, I can reduce the effect of outliers by applying log transformation and simply removing outliers, and I also can reduce the noise by selecting only important features. Moreover, the base classifier (Random Forests) is robust to outliers and noise. Therefore, AdaBoosted Random Forests is capable of overcoming the disadvantage of Adaboost and improving the result outputted by base classifier.

The AdaBoost I will use is the one in sklearn (*15). I will use the model I created by RandomForests as the base_estimator and adjust the parameters of n_estimators and learning_rate by GridSearchCV (*13). I also will input various values to the parameter of random_state to make the results stable.

2.4 Benchmark

To measure the perfromance of a solution model, I made a benchmark model and test the performance. Since it is a classification problem and there are many input variables, I'd like to try to use Gaussian Naive Bayes which needs less computation and works well with many features. I used sklearn's GaussianNB (*16) with the default parameters.

To do the benchmark, I applied following minimum preprocess. (For the codes, please see "first_refined_csv_maker_for_benchmark.py", "/lib/pre_process.py", "columns_config.py","benchmark.py", "benchmark_train_and_measure_performance" method in "train_and_test.py")

1.1, Inserting appropriate values for missing values of each feature (Imputation).
I could remove the lines which have missing values, but it wastes a lot of data because there are a lot of missing values, so I decided to apply following imputation to missing values. (See "first_refined_csv_maker_for_benchmark.py", "/lib/pre_process.py", "columns_config.py")

Table 8 – Imputation I applied for missing values.

The values to convert from	**The values to convert to**
NaN values in discrete features	94
NaN values in continuous features	The mean value for each feature.
The values in continuous features which represent "Refused", "Don't know" or "Not applicable" such as "9998", "9999", "98", "99", "996", "998", "999"	The mean value for each feature
"97" which represent "LESS THAN ONE" in **q29a**, **q29b** and **q29c**.	0.5

1.2 Applying value conversion to the target variable

I applied value conversion to the target variable as **Table 7** in the section of **Data Exploration**. (See "first_refined_csv_maker_for_benchmark.py", "/lib/pre_process.py", "columns_config.py".)

1.3, Adding new features

I added new features in the **Table 4** in the section of **Data Exploration.** (See "first_refined_csv_maker_for_benchmark.py".)

1.4, One-hot encoding.

I experimentally made the models with GaussianNB without applying one-hot encoding, but they scored badly on F1 score. The average score on F1 score with 10 different inputs is 0.213799. Since random guessing of F1 score is 0.1, the model seems not to be good enough as a benchmark model. For the reason, I applied one-hot encoding. (See "columns_config.py", "benchmark_train_and_measure_performance" method in "train_and_test.py", "/lib/pre_process.py",.)

Table 9 – The F1 scores of benchmark models without one-hot encoding with 10 different inputs.

random_state or Avg	F1 score with macro avg
0	0.227343
1	0.257149
2	0.178723
3	0.165101
4	0.270864

5	0.250733
6	0.189857
7	0.224883
8	0.177258
9	0.196082
Avg	0.213799

2, Result

I trained and tested 10 different inputs using different random states.
The average F1 score of the benchmark models is 0.444668 (**Table 10**). Since this is 4-class classification problem, the models perform much better than random guessing, so the benchmark models seem to be valid.

Table 10 – The F1 score of benchmark models with 10 different inputs. (To obtain this result, please execute "first_refined_data_maker_for_benchmark.py" first. Then execute "benchmark.py")

random_state or Avg	F1 score
0	0.462455965
1	0.509170654
2	0.424060595
3	0.442954602
4	0.405684506
5	0.474485101
6	0.416260497
7	0.413444123
8	0.460781804
9	0.437386621
Avg	0.444668447

3 Methodology

3.1 Data Preprocessing

I applied following data preprocess. The most of preprocesses are done in "first_refined_csv_maker.py". The step **4** and **5** are done in the file of "train_and_test.py"

1, Applied the same preprocesses as the ones in <u>Benchmark</u> section

In this step, I applied the same preprocesses as the ones in **<u>Benchmark</u>** section except for one-hot encoding. (See “first_refined_csv_maker.py”, “/lib/pre_process.py”, “columns_config.py”) One-hot encoding will be applied in the step **4** in this section.

2 Converting the data types of certain columns whose data types are float though they don’t need to be float.

Some one-hot encoded columns’ names contained float values such as '**q8_1.0**', '**q8_2.0**', '**q8_3.0**' even the original columns are categorical and don’t contain in any numbers except for 0 after the decimal points. Though it doesn’t affect the performance of the model, they look not cool. Just for the reason, I converted the data types from float64 to int64 and modified the one-hot encoded columns’ names. (converted '**q8_1.0**', '**q8_2.0**', '**q8_3.0**' to '**q8_1**', '**q8_2**', '**q8_3**') (See “first_refined_csv_maker.py”, “/lib/pre_process.py”, “columns_config.py”)

3, Applying log transformation

3.1, Raising negative values to positive

Since it is not possible to apply log transformation to negative values, I need to raise negative values to positive values while keeping the difference between each value. I achieved that by calculating the value which makes the smallest value of each feature to 1 and adding it to all the values in each feature. (See “first_refined_csv_maker.py”, “/lib/pre_process.py”, “columns_config.py”)

3.2 Applying log transformation

To reduce the negative effect to learning algorithms by highly skewed data, I decided to apply log transformation to the features whose absolute skew is more than 1. The absolute skew of each feature is mainly calculated by scipy.stats.skew (* 8). (See “first_refined_csv_maker.py”, “/lib/pre_process.py”, “columns_config.py”)

The reason I decided it to 1 is the value performed the best with my initial solution among the values between 0 to 4.5, and the value was expected to make the final solution perform well.

Table 11 - The result of experiment that measures the performance of each model with 10 different inputs and 10 different ‘allowed_skew’ which is a parameter of

“PreProcess.set_log_trans_based_on_abs_skew” (see “/lib/pre_process.py”). “Avg Accuracy of RF” represents the average F1 score of classifier made of Random Forests with 10 different inputs. “Avg Accuracy of ARF” represents the one made of AdaBoosted Random Forests with 10 different inputs. (To obtain those results, you have to manually input each value to the parameter of “pre_process_obj.set_log_trans_based_on_abs_skew” method, and excute “first_refined_csv_maker.py”, then execute “initial_solution.py”.)

Allowed Skew	Avg F1 score of RF	Avg F1 score of ARF	Avg F1 score of these two
0	0.4678	0.3219	0.39485
0.5	0.4306	0.3311	0.38085
1	0.4809	0.3728	0.42685
1.5	0.4596	0.3552	0.4074
2	0.4372	0.3476	0.3924
2.5	0.4467	0.3366	0.39165
3	0.4484	0.3365	0.39245
3.5	0.4678	0.3164	0.3921
4.0	0.4696	0.3375	0.40355
4.5	0.4429	0.3517	0.3973

For the skews of each feature, please see the **Fig 2** in **Data Explorlation** section. That way, you can see what features I applied log transformation.

4, Appling one-hot encoding

I applied one-hot encoding to the input variables. (See “train_and_test.py”, “pre_process.py”, “columns_config.py”)

5, Removing Outliers

Since AdaBoost is prone to be sensitive to outliers, I detected the outliers by “LocalOutlierFactor” in sklearn (*17) and removed them from training dataset. (See “train_and_test.py”)

3.2 Implementation

1, Implementation for making machine learning models.

I implemented the codes for making the machine learning models in the following steps.

1.1, PreProcess class.

I implemented “PreProcess” class in ‘/lib/pre_process.py’ with its unittests in “/tests/test_pre_process.py”. It can handle the most of preprocess I need by just inputting the configuration into “columns_config.py”. To understand the detail of how this “PreProcess” class works, please see the file of “/lib/pre_process.py”. Here is the list of the methods of this class.

Table 12 – Methods of “PreProcess” class.

Method name	**Description**
__init__	Constructor of the class. Requires pandas dataframe (self.df) which you wish to apply preprocess and the configuration of how you wish to apply preprocess (self.columns_config).
extract_data	Extracts columns you specified in columns_config from the dataframe(self.df)
average	Gets the average of specified column in self.df.
set_one_hot_encoding_list	Extracts columns which you want to apply one-hot encoding in the form of a list and stores it to self.one_hot_encoding_list.
apply_one_hot_encoding	Applies one-hot encoding to self.df's columns according to self.columns_config.
set_target_column_names	Finds target variables from self.columns_config and stores the column names to self.target_column_names.
get_inputs_and_outputs	Divides input and output variables and returns them

drop_columns_by_regexp	Drops matched columns by regular expressions from self.df
get_matched_column_names	Gets matched columns from self.df by regular expressions.
get_column_names_to_apply_log_trans	Gets column names which you specified to apply log transformation
apply_log_trans_according_to_columns_config	Applies log transformation to self.df accoording to self.columns_config
apply_log_trans	Applies log transformation to specified columns in self.df.
convert_values	Converts values in each column according to columns_config.py,
convert_types	Converts types of values of each column according to columns_config.py,
get_column_names_contain_continuous_values	Gets names of columns which contain continuous values according to columns_config.
get_skews	Gets each skew of continuous variables.
check_if_columns_contain_negative_values	Checks if continuous columns have negative values or not
raise_values_to_positive	Raise negative values to positive in order to apply log transformation
set_log_trans_based_on_abs_skew	Sets "'log trans' = True" to self.columns_config based on each abs skew of each column to apply log transformation

1.2, The configuration of preprocess

I wrote a configuration of preprocess in “columns_config.py”. This configuration can specify which columns you wish to extract. how the missing values in each column should be converted to, if a column contains continuous values or not, if you wish to apply one-hot encoding to a column or not and so on. For the detail of the configuration, please see “columns_config.py” and ‘/lib/pre_process.py’

1.3, First preprocess to the dataset.
I wrote the code which actually applies specified preprocesses to each column according to "columns_config.py" except for one-hot encoding and removing outliers.

Table 13 - Descriptions of the files which actually apply specified pre processes to each column according to "columns_config.py" except for one-hot encoding and removing outliers

File name	Description
first_refined_csv_maker.py	Applies preprocess except for one-hot encoding and removing outliers and outputs "first_refined_data.csv" which is necessary for actual machine learning process in "train_and_test.py".
first_refined_csv_maker_for_benchmark.py	Applies preprocess for benchmark except for one-hot encoding and removing outliers and outputs "first_refined_csv_for_benchmark.csv" which is necessary for making benchmark models by "benchmark.py" .

1.4, Second preprocess to apply to the first preprocessed dataset.
I wrote the function which applies second preprocess to the first preprocessed dataset according to "columns_config.py". The code is written in "train_and_test.py".

Table 14 - Descriptions of the function which applies second preprocess to the first preprocessed dataset according to "columns_config.py"

Function name	**Description**
get_feature_final_and_target_var	Loads the first preprocessed dataset such as "first_refined_data.csv" and "first_refined_csv_for_benchmark.csv", divides them into input and target variables and applies one-hot encoding. It outputs second-preprocessed input

	variables and target variables which are necessary for functions of training and testing models in “train_and_test.py”

1.5, Functions which train machine learning models and measure their performances
I wrote the functions which train models and measure the performances in “train_and_test.py”. Here are the descriptions of the functions.

Table 15- Descriptions of the functions which train models and measure the performances in “train_and_test.py”.

Function name	**Description**
initial_train_and_measure_performance	Trains initial solution models made by Random Forests and AdaBoosted Random Forests, measures their performances.
train_and_measure_performance_1	Trains models made by Random Forests and AdaBoosted Random Forests and measures their performances while applying Grid Search.
train_and_measure_performance_2	Trains models made by Random Forests and AdaBoosted Random Forests with specified parameters and measures their performances. The first model made by Random Forests can apply feature selection.
train_and_measure_performance_3	Trains models made by Random Forests with specified parameters and measures their performances. The first model made by Random Forests can apply feature selection.
benchmark_train_and_measure_performance	Trains benchmark models made by Gaussian Naïve Bayes with specified parameters and measures their performances.

1.6, Functions which make Training and testing with different random_states and showing and summarizing the results easy.

To check the robustness of the models, I had to run the process of training and testing multiple times with different inputs. However, it takes a lot of time to do that by manually inputting different random_states, so I made “generate_results” method in “/lib/results_handler.py” to make it easy. Moreover, to make showing machine learning result and summarizing the result easy, I also wrote “showing_mean_values” method and “summurize_results” method in the same file.

Table 16– Functions in “/lib/results_handler.py”.

Function name	**Description**
generate_results	Executes specified function of training and testing the model with different random_states and returns the results
showing_mean_values	Prints mean values of specified columns in given dataframe.
summurize_results	Summarizes machine learning results according to specified configuration. Currently, it only summarizes mean values of specified columns

2, Complications that occurred during this coding process

Codeing “PreProcess” class is the most time consuming part because I had to apply a lot of preprocess due to the algorithms and dataset I use. I had to write a lot of functions especially regarding log transformation and converting values. Moreover, it takes time to check the behavior of the methods in “PreProcess” class with the actual dataset, and the methods have to work precisely, because if I make mistakes in the preprocess, I cannot make good machine learning models. For the reasons, I wrote unittests for “PreProcess” class. That way, I can check the behaviors of the methods quickly and guarantee the performance of the methods at a certain level.

3.3 Refinement.

(**Note**: From this section, we need the file named “first_refined_data.csv” which contains preprocessed dataset, so if you want to get the same result as I get and there are no “first_refined_data.csv”, please execute “first_refined_csv_make.py” and make “first_refined_data.csv” first.)

The initial solution models are the ones made by Random Forests and the ones made by AdaBoosted Random Forests with their default parameters (See “initial_train_and_measure_performance” in “train_and_test.py”). **Table 17** shows the F1 scores of initial solution models with 10 different inputs.

Table 17 – F1 scores of initial solution models made by Random Forests (RF) and AdaBoosted Random Forests (ARF) with 10 different inputs. (To obtain this result, execute “initial_solution.py”.)

random_state or avg	F1 score of base classifier (RF)	n_estimators of base classifier (RF)	F1 score of boosted classifier (ARF)	n_estimators of boosting classifier (ARF)	learning_rate of boosting classifier (ARF)
0	0.5508	10	0.3999	50	1.0
1	0.4833	10	0.4012	50	1.0
2	0.3986	10	0.3325	50	1.0
3	0.4491	10	0.4336	50	1.0
4	0.5284	10	0.4264	50	1.0
5	0.5378	10	0.3057	50	1.0
6	0.4155	10	0.4500	50	1.0
7	0.3917	10	0.3636	50	1.0
8	0.5029	10	0.3297	50	1.0
9	0.5513	10	0.2857	50	1.0
avg	0.4809	10	0.3728	50	1.0

Adjusting parameters of algorithm

To find the best parameters of the models, I did grid search by using sklearn ‘s GridSearchCV (*13).

For the RandomForests, I let it select its n_estimators from 10,50,100,200,300,400,500. For AdaBoostClassifier, I let it select its n_estimators from 10, 25, 50 and its learning_rate from 0.05, 0.1, 0.15. I executed this grid search with the input which is generated by "random_state == 0".

Table 18 - The F1 score of the models made by Random Forests (RF) and AdaBoosted Random Forests (ARF) and the best parameters with the input which is generated by "random_state == 0". (To obtain the result, executes "refinement_1.py")

F1 score of base classifier (RF)	n_estimators of base classifier (RF)	F1 score of ARF	n_estimators of boosting classifier (ARF)	learning_rate of boosting classifier (ARF)
0.4864	300	0.4761	10	0.05

Then I checked the robustness of the model made by optimized parameter with 10 different inputs.

Table 19 - The F1 score of the parameter-adjusted models made by Random Forests (RF) and Adaboosted Random Forests (ARF) with 10 different inputs. (To obtain the result, executes "refinement_1_2.py" with "random_states = range(10) ".)

random_state or average	F1 score of base classifier (RF)	n_estimators of base classifier (RF)	F1 score of ARF	n_estimators of boosting classifier (ARF)	learning_rate of boosting classifier (ARF)
0	0.4864	300	0.4761	10	0.05
1	0.4770	300	0.4689	10	0.05
2	0.3816	300	0.3916	10	0.05
3	0.5124	300	0.4927	10	0.05
4	0.4693	300	0.4466	10	0.05
5	0.3987	300	0.4438	10	0.05
6	0.3786	300	0.3847	10	0.05
7	0.4226	300	0.4086	10	0.05
8	0.4192	300	0.4139	10	0.05
9	0.5525	300	0.4273	10	0.05
Avg	0.4498	300	0.4354	10	0.05

The average F1 score of the models made by Random Forests becomes 0.03111 lower than the one of the initial solution. On the other hand, the average F1 score of the models made by AdaBoosted Random Forests improved 0.062607 from the one of the initial solution.

Since I adjusted the parameters, the fact that the average F1 score of the models made by AdaBoosted Random Forests significantly improved from the one of the initial models totally makes sense. However, the performance of the base classifier has worsened though I applied grid search and adjusted the parameters. This is not easily acceptable result, so I also checked the performance of the initial solution models and parameter-adjusted models by 30 different inputs.

Table 20– The average F1 scores of the initial solution models and parameter-adjusted models with 30 different inputs. (To obtain this result, change "random_states = range(10) " to "random_states = range(30) " in the files of "initial_solution.py" and "refinement_1_2.py")

Type of models	Avg F1 score of base classifier (RF)	Avg F1 score of boosted classifier (ARF)	Avg F1 score of these two
Initial solution models	0.4558	0.3573	0.4066
Adjusted models	0.4296	0.4358	0.4327

Still the average F1 score of base classifier of initial solution models is better than the one of parameter-adjusted models, so I think I have to accept the fact that the initial solution models made by Random Forests performs better than the parameter-adjusted models made by Random Forests in this step.

Removing less important features

Since AdaBoost is prone to be sensitive to noise, I removed less important features and only reserved important features. To figure out what features are important, I used the "feature_importance_" property in skelearn's "RandomForestClassifier"(*12). I checked the performance of the initial solution models made by Random Forests and parameter-adjusted models made by AdaBoosted Random Forests with 10 different inputs.

Table 21– The average F1 score of feature-reduced models made by Random Forests (RF) and Adaboosted Random Forests (ARF) with 10 different inputs. I checked the average F1 score of reserving top 25, 50, 100, 150, 200 important features. (To obtain the result, set "'n_estimators': 10" in "other_args" variable in "refinement_2.py" and execute "refinement_2.py" and "refinement_2_2.py" while changing "f_select_range" of "other_args" variables to "[0,25]", "[0,50]", "[0,100]", "[0,150]" and "[0,200]".)

Number of top important features reserved	Avg Acc of feature-reduced RF	n_estimator of RF	Avg Acc of feature-reduced ARF	n_estimator of base classifier of ARF	learning_rate of boosting classifier of ARF	n_estimator of boosting classifier of ARF
25	0.4511	10	0.5177	300	0.05	10
50	0.4606	10	0.5087	300	0.05	10
100	0.4681	10	0.5107	300	0.05	10
150	0.4614	10	0.5001	300	0.05	10
200	0.4880	10	0.4791	300	0.05	10

The feature-reduced models made by AdaBoosted Random Forests with top 25 important features scored the best on average. However, the feature-reduced models made by Random Forests with "n_estimator = 10" didn't score well. The most of scores are worse than initial solution models made by Random Forests, so the models may not be robust enough to small changes in input features. For the reason, I also tried the parameter-adjusted feature-reduced models made by Random Forests whose "n_estimators" is 300.

Table 22 – The average F1 score of feature-reduced models made by Random Forests (RF) whoe "n_estimator" is 300 with 10 different inputs.　I checked the average F1 score of reserving top 25, 50, 100, 150, 200 important features. (To obtain the result, set

“'n_estimators': 300” in “other_args” variable in "refinement_2.py" and executes "refinement_2.py" while changing “f_select_range” of “other_args” variable to “[0,25]”, “[0,50]”, “[0,100]”, “[0,150]” and “[0,200]”.)

Number of top important features reserved	Avg Acc of feature-reduced RF	n_estimators of RF
25	0.5355	300
50	0.5229	300
100	0.4975	300
150	0.4671	300
200	0.4728	300

The average F1 score of feature-reduced models made by Random Forests (RF) whose “n_estimator” is 300 scores better than the ones whose “n_estimator” is 10. Since further refinement doesn’t seem to improve the performance significantly, I finish the refinement as this level. Here is the summary of improvement

Table 23 – The summary of improvement of the models with 10 different inputs. RF = Random Forests. ARF = AdaBoosted Random Forests.

Step	Avg Acc of RF	Avg Acc of ARF
Initial Solutions	0.4809	0.3728
After optimizing parameters	0.4498	0.4354
After feature selection	0.5355	0.5177

For the models made by Random Forests, I achieved improvement of 0.054555 on average accuracy from the ones of the initial solution. For the models made by AdaBoosted Random Forest, I achieved improvement of 0.144874 from the ones of the initial solution.

4 Results

4.1 Model Evaluation and Validation

The final model I chose is the one made by Random Forests which uses only top 25 important features with optimized parameters. The n_estimators of the feature selecting model and final model is both 300. The reason I choose this model as a final model is that it averagely scores best with 10 different inputs and the model created by only using top 12 important features also performed stably (**Table 24, Table 25**), so the model seems to be robust enough.

Table 24 – The F1 score and computing time of the final model with 10 different inputs.(To obtain the result, execute "solution.py")

random_state or average	Acc of final model (RF)	n_estimators	Computing time
0	0.568744802	300	2.536000013
1	0.566295707	300	2.370000124
2	0.523586528	300	2.434999943
3	0.506136478	300	2.409000158
4	0.552813241	300	2.434000015
5	0.515694013	300	2.493999958
6	0.471668566	300	2.434000015
7	0.524880614	300	2.421000004
8	0.501182466	300	2.599999905
9	0.624174055	300	2.427999973
Avg	0.535517647	300	2.456100011

The final model seems to perform stably with various inputs though the model made by the inputs generated by "random_state == 6" seems to slightly underperform.

The average F1 score of the final model created by using only top 12 important features is 0.501740614. It is 0.033777033 lower than the one of the final model. I think this is acceptable worse and the result proves the model has the robustness that doesn't greatly vary the result by applying this amount of small changes to the inputs

Table 25 – The F1 score and computing time of the final model created by using only top 12 important features with 10 different inputs. (To obtain the result, execute

"check_solution_robustness_1.py")

random_state or average	Acc of final model with top 25 features(RF)	n_estimators	Computing time
0	0.454394329	300	2.277000189
1	0.556909576	300	2.255999804
2	0.525437687	300	2.221000195
3	0.443501326	300	2.269000053
4	0.531319441	300	2.312999964
5	0.537523593	300	2.282000065
6	0.430574790	300	2.269000053
7	0.444006509	300	2.256999969
8	0.577917131	300	2.261000156
9	0.515821761	300	2.229000092
Avg	0.501740614	300	2.263400054

4.2 Justification

The average F1 score of the final model with 10 different inputs is 0.535517647, and the one of the benchmark model made by Gaussian Naïve Bayes is 0.444668447. The score of the former is much better than the latter. I can say I succeed to make the solution model which performs better than the benchmark model.

Since this problem is 4-class classification problem, the F1 score of predicting target variable by random guessing is 0.1 (execute experiment_metrics_2.py). Since the F1 score of the final model is 0.535517647 on average, it performs 5.35517647 times better than the prediction of random guessing. Though it is not a perfect model, the important features extracted from the model with this quality has fairly good persuasion to tell what brings us good sleep

5 Conclusion

5.1 Free-Form Visualization

Since our aim is to figure out the factors which are capable of improving our sleep quality, I will use this free-form section for its analysis.

First of all, I will provide a table which represents top 30 important features and their correlations with **q30** of final model with top 50 important features. The reason I used final model with top 50 important features is the one with top 25 important features doesn't have some of important features which give us useful hints to improve our sleep quality. Moreover, the model with top 50 important features still scores 0.5229 on F1 score. It is just 0.0126 lower than the final model with top 25 important features, so I think using this model for this analysis still has fairly good persuasion to tell what brings us good sleep. The correlations between the features and sleep quality (**q30**) are calculated by using "pandas.DataFrame.corr" method in pandas (*18).

Table 26 – Top 30 features of final model with top 50 important features which are important to predict self-reported sleep quality (**q30**) and correlations between those features and self-reported sleep quality (**q30**). (To obtain the results, execute "important_features_of_final_model.py".)

Feature name	Feature importance	Correlation with sleep quality(**q30**)
BMI	0.035398883	-0.091856238
q11_2	0.030108686	-0.430447905
q11_4	0.029754460	0.300168374
q11_3	0.028715518	0.411062741
EPWORTH	0.028055642	-0.197462768
Q43TOTAL	0.027905315	-0.082249173
q19a_4	0.026521181	-0.461272943
IPAQTOTAL	0.026495448	0.061283362
Q36Q38Q40TOTAL	0.025150021	0.023762434
qs1	0.025147771	0.018571938
Q5	0.023587141	0.289930251
Q43B	0.023568818	-0.020239368

Q6	0.023198969	0.254857633
Q4Q3DIF	0.023136144	0.117905513
q12_2	0.022710189	-0.432559454
Q42	0.022703489	-0.089904238
HEIGHT	0.022580160	0.149404082
Q2Q1DIF	0.022170605	0.053910396
Q4VALUE	0.021959532	0.081949336
IPAQ40	0.021548239	0.041304327
Q43E	0.021395689	-0.036341292
WEIGHT	0.021230389	-0.038582263
NSFDISABLE	0.020825576	-0.299511399
q35	0.020771555	-0.072962482
Q40	0.020148239	0.035447910
Q6Q5DIF	0.019169573	0.002092796
Q4Q2DIFQ3Q1DIFTOTAL	0.018632387	-0.004660851
Q29TOTAL	0.018516235	-0.147795701
Q1VALUE	0.018309518	-0.070468674
Q3VALUE	0.017959109	-0.134121768

Fig 15 - Feature importance and correlation with **q30** of final model with top 50 important features.(Only shows the ones of top 1 to 15 important features)

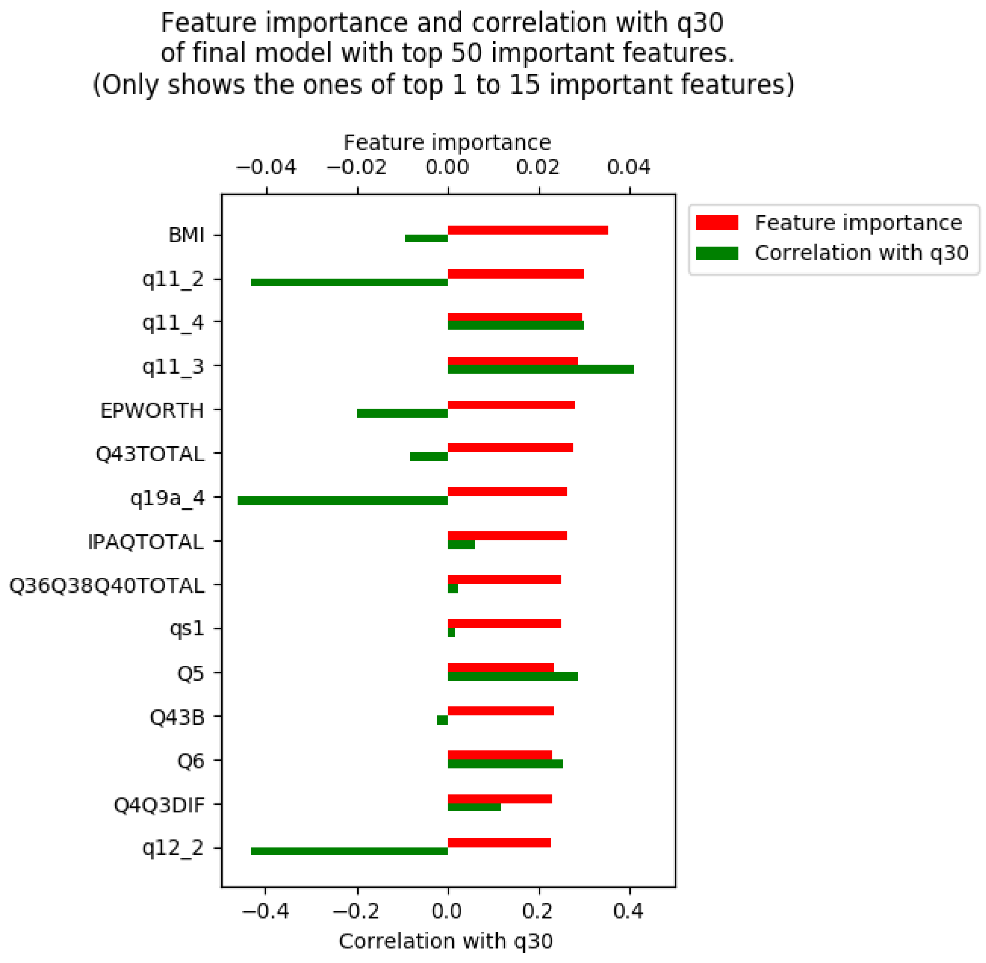
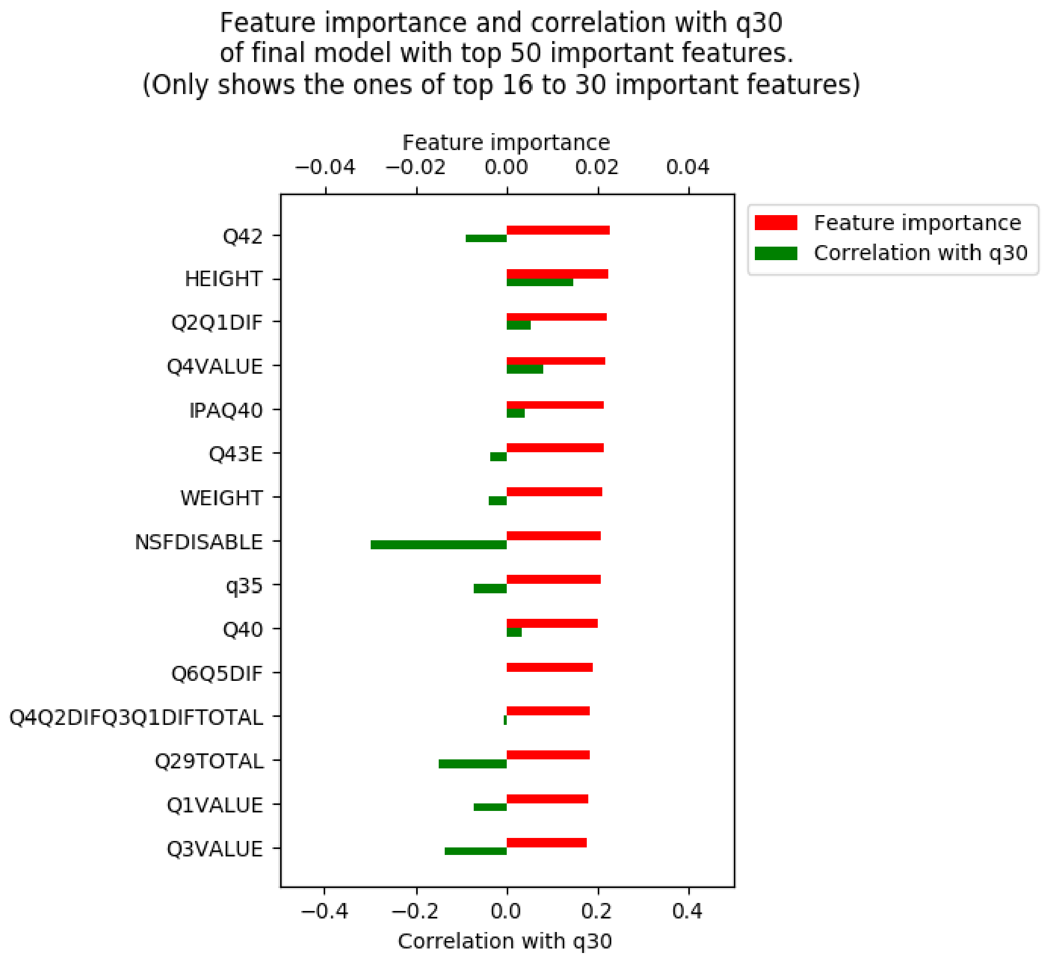


Fig 16 - Feature importance and correlation with **q30** of final model with top 50 important features.(Only shows the ones of top 16 to 30 important features)



The features which I think give useful hints to improve our sleep quality are following

IPAQTOTAL. Q43TOTAL ,Q36Q38Q40TOTAL, Q42, Q29TOTAL

Since this project aims to find out the features which improve the sleep quality of people who cannot sleep well though they take enough time for sleep, I don't list the features which represent length of sleep.

Doing physical activities may improve sleep quality

IPAQTOTAL and **Q36Q38Q40TOTAL** represent the respondents' amount of exercise, and those features are positively correlated with the self-reported sleep quality (**q30**).

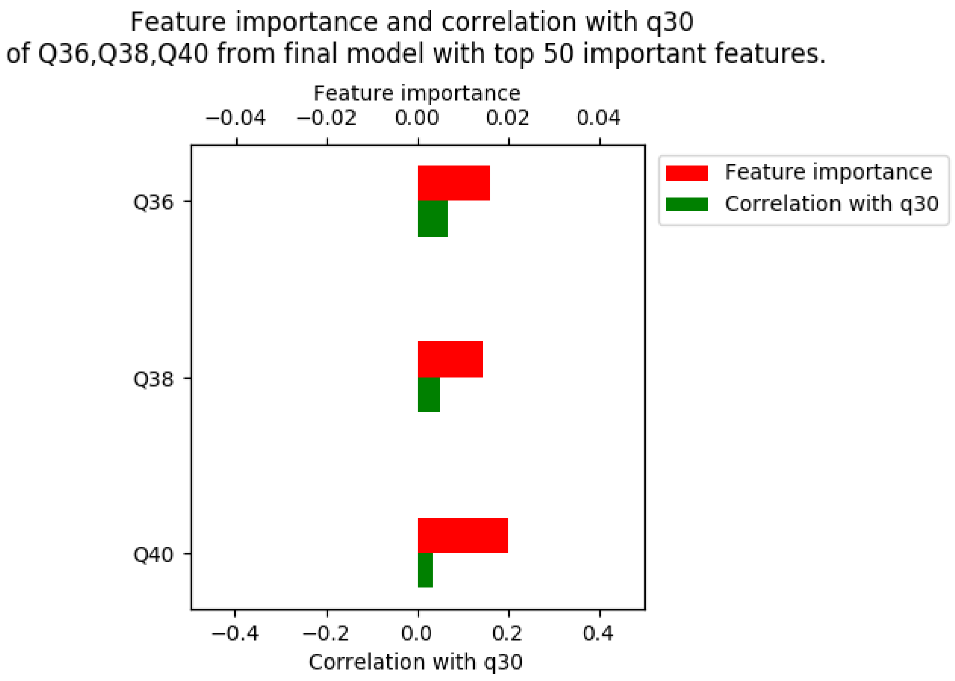
IPAQ is the abbreviation of INTERNATIONAL PHYSICAL ACTIVITY QUESTIONNAIRE which estimates a respondent's physical activity by his or her self-report (*19). The data collected from IPAQ can be reported as continuous scores (*20). The higher the scores, the greater amount of physical activity is. **Q36Q38Q40TOTAL** also represents the respondents' amount of physical activity (**Table 4**). Based on the result, I can make a hypothesis that physical activity can improve sleep quality.

I also tried to figure out which intensities of physical activity are the most efficient to improve sleep quality from the dataset. The feature which is correlated most positively with sleep quality (**q30**) among vigorous (**Q36**), moderate (**Q38**), light (**Q40**) physical activities is vigorous physical activities (**Q36**). Thus, among these intensities of physical activities, doing vigorous physical activities may be the most efficient to improve sleep quality.

Table 27 – The feature importance and correlation with sleep quality (**q30**) of vigorous, moderate, light physical activity (**Q36, Q38, Q40**)

Feature name	**Feature importance**	**Correlation with sleep quality(q30)**
Q36	0.016155009	0.068746574
Q38	0.014551969	0.053297491
Q40	0.020148239	0.035447910

Fig 17 - Feature importance and correlation with q30 of Q36,Q38,Q40 from final model with top 50 important features.



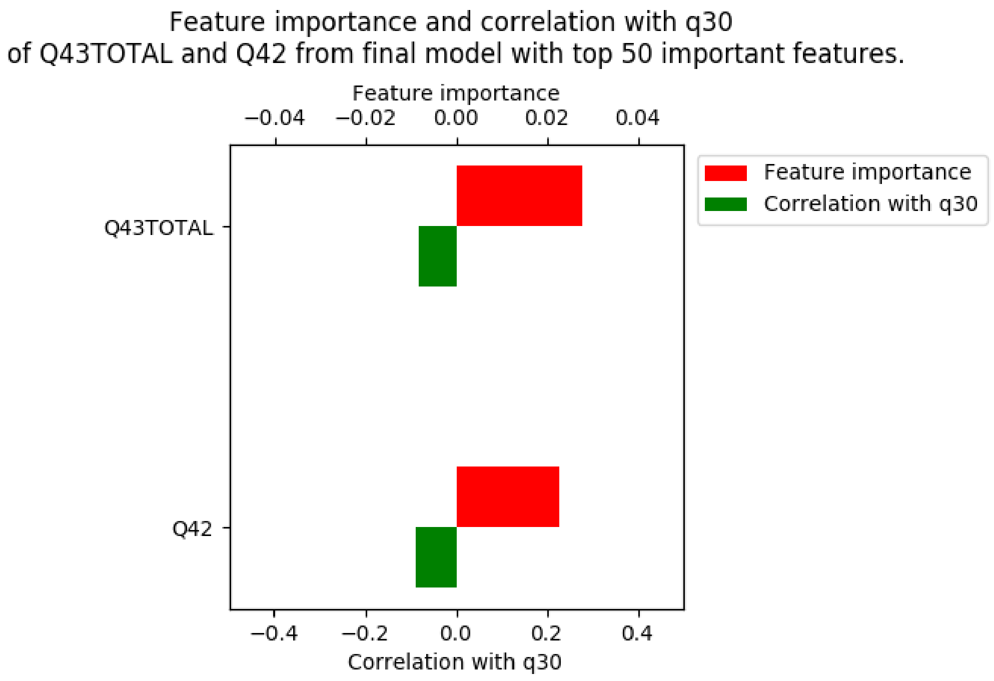
<u>**Sitting activity may worsen sleep quality**</u>

Q43TOTAL and **Q42** represent the time respondents spent for sitting in the past 7 days. These features are negatively correlated with the self-reported sleep quality (**q30**). Based on the result, it may be better to avoid sitting for long hours to improve sleep quality.

Table 28 – The feature importance and correlation with sleep quality (**q30**) of the features that represent amount of sitting time (**Q43TOTAL, Q42**)

Feature name	Feature importance	Correlation with sleep quality(**q30**)
Q43TOTAL	0.027905315	-0.082249173
Q42	0.022703489	-0.089904238

Fig 18 - Feature importance and correlation with Q43TOTAL and Q42 from final model with top 50 important features.



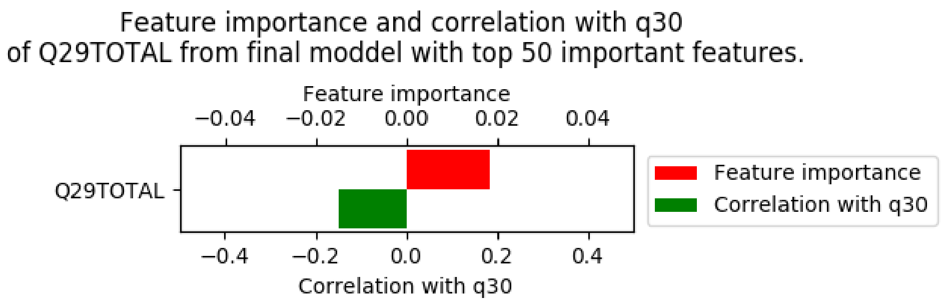
<u>**Drinking caffeinated beverages may worsen sleep quality**</u>

Q29TOTAL represents the amount of caffeine the respondents took per day in the last 2 weeks. This is strongly and negatively correlated with the self-reported sleep quality (**q30**). Based on the result, it may be better to avoid drink caffeinated beverages to improve our sleep quality though they are useful to keep us awaking during day time.

Table 29 – The feature importance and correlation with sleep quality (**q30**) of the features that represent amount of sitting time (**Q29TOTAL**)

Feature name	Feature importance	Correlation with sleep quality(**q30**)
Q29TOTAL	0.018516235	-0.147795701

Fig 19 - Feature importance and correlation with **Q29TOTAL** from final model with top 50 important features.



I also analyzed which period of time people shouldn't drink caffeinated beverages to improve sleep quality by observing these three features.

Table 30 – The descriptions of **q29a, q29b, q29c.**

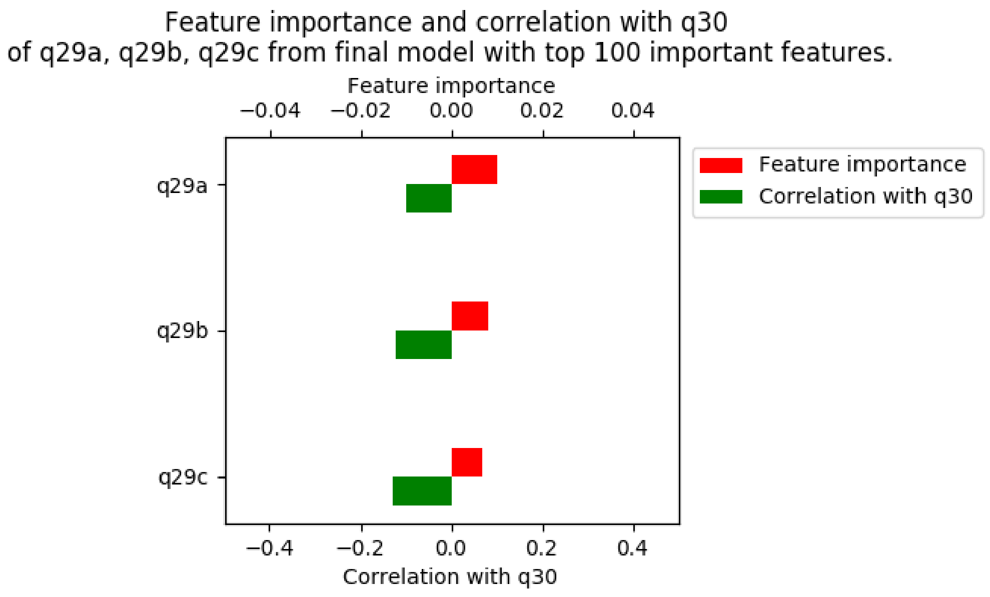
Feature name	**Description**
q29a	Number of 12 ounce servings of caffeinated beverages they take between 5:00 AM and noon.
q29b	Number of 12 ounce servings of caffeinated beverages they take between noon and 5:00 PM.
q29c	Number of 12 ounce servings of caffeinated beverages they take between 5:00 PM and 5:00 AM the next morning.

Since there aren't **q29a**, **q29b**, **q29c** together in the final model made by top 50 important features, I made the final model with top 100 important features which still scores 0.4975 on average F1 score with 10 different inputs and extracted those features' feature importance and correlation with sleep quality (**q30**) and show them on **Table 31**. Based on the table, **q29c** (Between 5:00 PM and 5:00 AM the next morning) is the most negatively correlated with sleep quality (**q30**) among those three, and **q29a** (Between 5:00 AM and noon) has the weakest negative correlation with sleep quality (**q30**). Since a lot of people sleep at night (See **Fig 20**, **Fig 21**) it totally makes sense.

Table 31– The feature importance and correlation with sleep quality (**q30**) of the features which represent the amount of caffeinated drink the respondents took during each time period (**q29a**, **q29b**, **q29c**) (To obtain this result, set 'f_select_range' of "other_args" variable in "important_features_of_final_model.py" to "[0,100]" and execute "important_features_of_final_model.py".)

Feature name	Feature importance	Correlation with sleep quality(**q30**)
q29a	0.010096343	-0.099883996
q29b	0.008081634	-0.122983230
q29c	0.006980736	-0.129496479

Fig 22 - Feature importance and correlation with **q29a**, **q29b**, **q29c** from final model with top 100 important features.



Summary –

1、 **Doing physical activities may improve sleep quality.**

2、 **Sitting activity may worsen sleep quality.**

3、 **Drinking caffeinated beverages may worsen sleep quality, especially drinking them at the time that is close to going to bed seems to be bad.**

Since the data is highly skewed, to answer the questions like "How much should we do physical activity?", "How much of sitting activity is acceptable?", "How much can we drink caffeinated beverages?", we need further research.

Some of interesting characteristic of sleep.

This is little bit out of main purpose of this project, however, I found some interesting characteristics of sleep through this project, so I will mention about it.

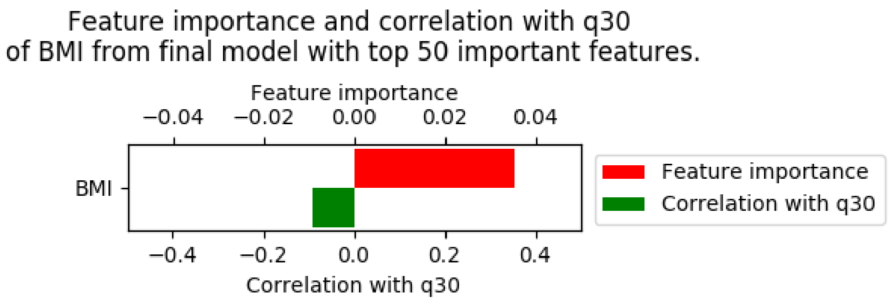
Fat people tend to sleep not well?

The feature importance of **BMI**(Body Mass Index) to predict self-reported sleep quality (**q30**) is 0.035398883. The correlation between **BMI** and self-reported sleep quality (**q30**) is -0.091856238. From this analysis, **BMI** is negatively correlated with self-reported sleep quality. However, to figure out the casual relationship between sleep quality and BMI, we need additional research.

Table 32– The feature importance and correlation with sleep quality (**q30**) of **BMI** from final model with top 50 important features.

Feature name	Feature importance	Correlation with sleep quality(**q30**)
BMI	0.035398883	-0.091856238

Fig 23 - Feature importance and correlation with **BMI** from final model with top 50 important features.



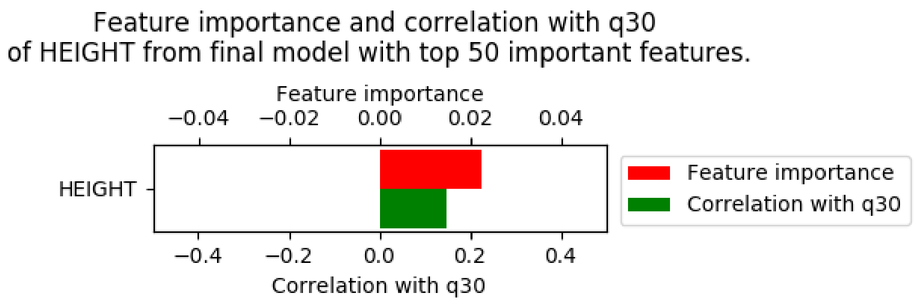
Height and sleep quality are positively correlated

The feature importance of **HEIGHT** to predict self-reported sleep quality (**q30**) is 0.02258016. The correlation between **HEIGHT** and self-reported sleep quality (**q30**) is 0.149404082. From this analysis, **HEIGHT** is relatively strongly positively correlated with self-reported sleep quality. As a supposable reason for this, I guess people who felt sleep well in the past two weeks also tended to feel sleep well in their growth period, so their heights became taller and the result shows the relatively strong positive correlation between **HEIGHT** and sleep quality (**q30**). However, to figure out the accurate casual relationship between heights and sleep quality, we need additional research.

Table 33 – The feature importance and correlation with sleep quality (**q30**) of **HEIGHT** from final model with top 50 important features.

Feature name	Feature importance	Correlation with sleep quality(**q30**)
HEIGHT	0.02258016	0.149404082

Fig 24 - Feature importance and correlation with **HEIGHT** from final model with top 50 important features.



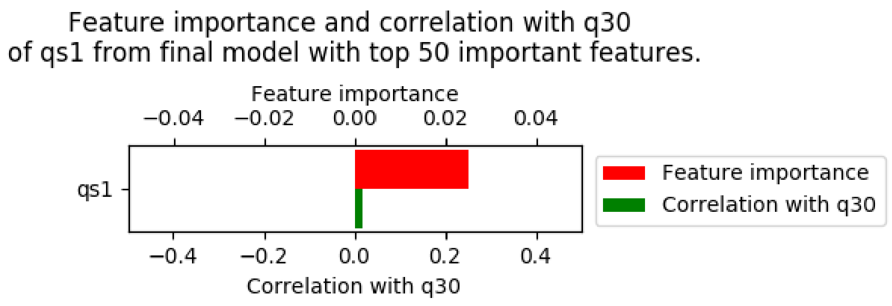
<u>**Age is one of the most important features to predict sleep quality**</u>
The feature importance of age (**qs1**) to predict self-reported sleep quality (**q30**) is 0.025388135. The correlation between age (**qs1** and self-reported sleep quality (**q30**) is 0.018571938. Based on the result, I can say age (**qs1**) is relatively very important feature to predict sleep quality among the input features, but I cannot see any strong correlation between ages (**qs1**) and sleep quality (**q30**). I guess age (**qs1**) and some features may be strongly correlated with sleep quality (**q30**), however,, to know what actually happens here, we need deeper analysis.

Table 34 – The feature importance and correlation with sleep quality (**q30**) and age(**qs1**)

Feature name	Feature importance	Correlation with sleep

		quality(**q30**)
qs1	0.025147771	0.018571938

Fig 25 - Feature importance and correlation with **qs1** from final model with top 50 important features.

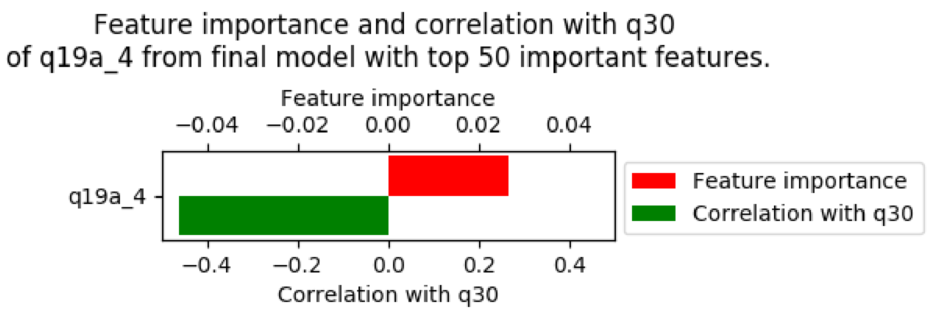


Moreover, we can see slight positive correlation between age (**qs1**) and sleep quality (**q30**). According to National Sleep foundation, as people age they tend to have a harder time falling asleep and more trouble staying asleep than when they were younger(*21). From my project, the feature of **q19a_4** which represents a respondent had difficulty with falling asleep every night or almost every night is strongly and negatively correlated with sleep quality (**q30**) (See **Table 35**). Thus, it is natural to guess that old people in this dataset tend to not sleep well too, but it doesn't happen here. I guess old people in our dataset are not old enough to get low-quality sleep or older people in this dataset may have some hints to overcome old people's low-quality sleep.

Table 35– The feature importance and correlation with sleep quality (**q30**) and whether the respondents had difficulty falling asleep every night or almost every night in the past two weeks.(**q19a_4**)

Feature name	Feature importance	Correlation with sleep quality(**q30**)
q19a_4	0.026521181	-0.461272943

Fig 26 - Feature importance and correlation with **q19a_4** from final model with top 50 important features.



5.2 Reflection

Summary of the entire process of this project

This project is processed in following steps

1. Looking for the problem I am interested in while searching a valid dataset on the internet.
2. Choosing the problem and the dataset which can solve the problem.
3. Studying more about background of the problem and the features of the dataset while considering the appropriate machine learning algorithm and making and viewing visualizations of the dataset.
4. Selecting features to use for making models and deciding preprocess to apply for the dataset based on the research in 3.
5. Searching for the some processes which can be used for the preprocess in some libraries such as numpy, pandas, sklearn and so on, then designing and coding a class for the preprocess with unittests.
6. Applying preprocesses to the dataset.
7. Coding and trying different models while adjusting the model's parameters.

8, Analyzing the result.

Interesting aspects of the project

Just looking the dataset and the machine learning result is interesting for me. These kinds of statistical information give me new discoveries and often break my preconception.

<u>Difficult aspects of the project</u>
The coding part was hard for me because I am kind of new to python and coding for machine learning. I had to go back previous steps many times because there were a lot of unexpected behaviors of the codes and unexpected needs from the machine learning models and the analysis.

<u>Does the final model and solution fit your expectations for the problem?</u>
The solution model improved to the quality which can somehow tell what brings us good sleep though it's not perfect, but I hesitate to use it as a general setting to solve these types of problems. I defined this problem as a multi classification problem. However, later I realized it could be converted to a regression problem. I think if I defined it as a regression problem, the model would tell people's sleep quality more precisely. In this multi classification problem, the model only can predict 1,2,3 and 4. However, if I convert it to a regression model, the model can output the values which are located between those classes such as 2.423, 1.875, 3.229 and so on. In other words, multi classification only can use four colors to draw a picture. However, regression can use countless number of colors to draw a picture. It is obvious which can draw a more realistic picture.

<u>5.3 Improvement</u>

<u>Further improvements that could be made</u>
As I said in the **<u>Reflection</u>** section, converting this problem to regression problem is the one I could try, because the model's prediction could be more precise.

Even if I keep the problem as a multi classification problem, I think adjusting the parameters more closely could improve the performance of the model more, however, I cannot find out the way to improve the performance of the model significantly.

<u>Citations</u>

*1, “Missing just a couple of hours of sleep doubles car crash risk” from CBS NEWS
https://www.cbsnews.com/news/sleep-deprivation-doubles-car-crash-risk-aaa/
*2, “Sleep and Disease Risk” from Healthy Sleep
http://healthysleep.med.harvard.edu/healthy/matters/consequences/sleep-and-disease-risk
*3, “Sleep, Learning, and Memory” from Healthy Sleep
http://healthysleep.med.harvard.edu/healthy/matters/benefits-of-sleep/learning-memory
*4 “Overview of Sleep Disorders” from healthcommunities.com
http://www.healthcommunities.com/sleep-disorders/overview-of-sleep-disorders.shtml
*5, “2013SleepinAmericaPollExerciseandSleepRawDataExcel” from National Sleep Foundation
http://www.sleephealthjournal.org/pb/assets/raw/Health%20Advance/journals/sleh/2013SleepinAmericaPollExerciseandSleepRawDataExcel.xls

---

*[1] “Sleep, Learning, and Memory” from Healthy Sleep
http://healthysleep.med.harvard.edu/healthy/matters/benefits-of-sleep/learning-memory

*[2] “Sleep and Disease Risk” from Healthy Sleep
http://healthysleep.med.harvard.edu/healthy/matters/consequences/sleep-and-disease-risk

*[3] “Missing just a couple of hours of sleep doubles car crash risk” from CBS NEWS
https://www.cbsnews.com/news/sleep-deprivation-doubles-car-crash-risk-aaa/

*[4] “Overview of Sleep Disorders” from healthcommunities.com
http://www.healthcommunities.com/sleep-disorders/overview-of-sleep-disorders.shtml

*[5] “2013SleepinAmericaPollExerciseandSleepRawDataExcel” from National Sleep Foundation
http://www.sleephealthjournal.org/pb/assets/raw/Health%20Advance/journals/sleh/2013SleepinAmericaPollExerciseandSleepRawDataExcel.xls

*[6] “2013 Sleep in America Poll - Exercise and Sleep - Summary of Findings” provided by National Sleep Foundation

https://sleepfoundation.org/sites/default/files/RPT336%20Summary%20of%20Findings%2002%2020%202013.pdf

*７ “2013 SLEEP IN AMERICA POLL: PHYSICAL ACTIVITY AND SLEEP SCREENING QUESTIONNAIRE” from National Sleep Foundation
https://sleepfoundation.org/sites/default/files/SIAQuestionnaire2013.pdf

*８ “scipy.stats.skew” from SciPy.org
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.skew.html

*９ “Aerobic exercise improves self-reported sleep and quality of life in older adults with insomnia” by Kathryn J. Reid, PhD, Kelly Glazer Baron, PhD, Brandon Lu, MD, Erik Naylor, PhD, Lisa Wolfe, MD, and Phyllis C. Zee, MD, PhD
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2992829/?tool=pubmed

*１０ ”How long are you affected by caffeine?” from ScienceNordic
http://sciencenordic.com/how-long-are-you-affected-caffeine

*１１ “Random Forest algorithm” from Machine Learning Madness
http://amateurdatascientist.blogspot.jp/2012/01/random-forest-algorithm.html

*１２ “sklearn.ensemble.RandomForestClassifier” from scikit-learn
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

*１３ “sklearn.model_selection.GridSearchCV” from scikit-learn
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

*１４ “Ada Boost” from NickGillianWiki
http://www.nickgillian.com/wiki/pmwiki.php/GRT/AdaBoost

*１５ “sklearn.ensenmble.AdaBoostClassifier” from scikit-learn
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

*１６ “sklearn.naive_bayes.GaussianNB” from scikit-learn.
http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

*１７ “sklearn.neighbors.LocalOutlierFactor” from scikit-learn.
http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html

*１８ “pandas.DataFrame.corr” from pandas 0.22.0 documentation
https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.corr.html

*１９ “INTERNATIONAL PHYSICAL ACTIVITY QUESTIONNAIRE”
http://www.sdp.univ.fvg.it/sites/default/files/IPAQ_English_self-admin_long.pdf

*２０ “Guidelines for Data Processing and Analysis of the International

Physical Activity Questionnaire (IPAQ) - Short Form”
http://www.institutferran.org/documentos/scoring_short_ipaq_april04.pdf

*２１ “AGING AND SLEEP” from National Sleep Foundation
https://sleepfoundation.org/sleep-topics/aging-and-sleep