A Systematic Exploration of Health Data from Wearable Devices and Implementation of A Machine Learning Model to Predict User's Sickness

Dr. Asif Iqbal

March 13, 2020

Contents

1 Abstract

The objective of this project is to utilize the health data acquired by wearable devices (*e.g.*, smart watch) to develop a predictive model based on machine learning (ML) algorithms. The final outcome of this model is to predict whether a specific user feels sick or not sick with the health data obtained from that user. This report outlines a detailed demonstration of how to achieve this goal, which is also summarized in Fig. 1. In this regard, we started our discussion by looking into the raw data and building our solid understanding of the data. Then we performed an elaborate exploratory data analysis including formulation of new features, imputation of missing value, statistical analysis and data visualization. This process delivered a clean dataset, which turned out to be extremely powerful to predict user's sickness. This project also tackled the critical issues originating due to the imbalance nature of our dataset. In this project, we developed 7 machine learning models and optimized each model by tuning the hyperparameters. Throughout our exhaustive analysis, we have developed the best performing machine learning model using an optimally designed Gradient Boosting classifier that can predict a user's sickness with an F1-score of 0.97 and a recall score of 0.96.

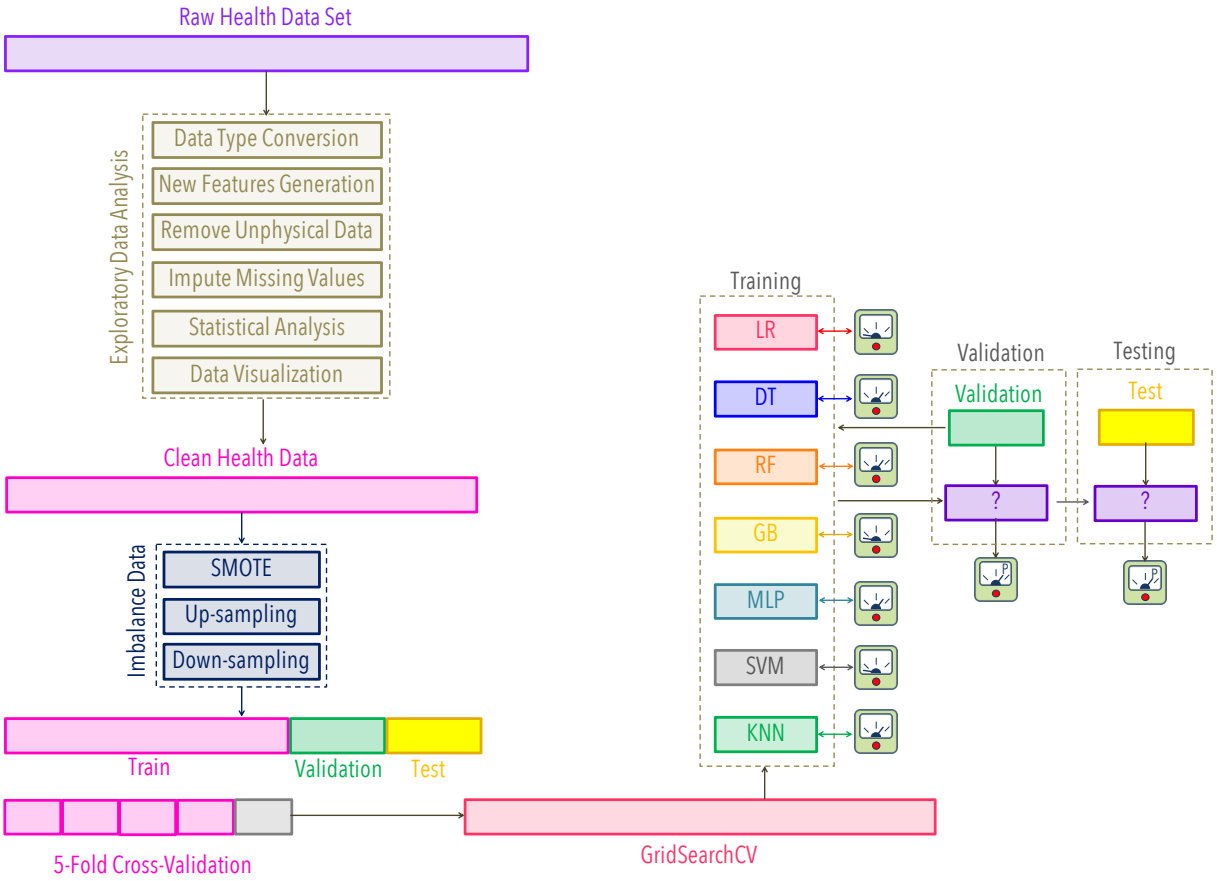

Figure 1: An overview of the process incorporated in this project, starting from the raw data to clean data generation, followed by the data upsampling (*via* SMOTE) to deal with the imbalance nature of the dataset, train/validation/test sets splitting and grid search based simultaneous model training and validation. At the end of this process, test dataset is utilized to conduct an unbiased evaluation of the performance of the developed models.

2 Data Importation and a Quick Look into Its Properties

With significant recent development in the smart and light-weight wearable technology, a large number of population can now afford and use wearable devices. The key aspect of this technology is to continuously monitor a user's health related metrics, which include but are not limited to, a daily count of steps, amount of physically active and inactive time, sleep metrics as well as environmental conditions (*e.g.*, temperature, humidity, air pressure). The wide adaptation of these devices has introduced numerous intriguing opportunities to utilize the power of the large amount of data acquired by wearable devices for further betterment of human lifestyle. For example, it would be exciting to use this continuous and abundant health data from wearable devices to predict a user's physical condition. In the same spirit, the project outlined in this report also utilizes wearable data from 243 anonymized users obtained in a period of a few months and excavates a machine learning model that can predict whether a user feels sick or not sick on a particular day. The very first look of our data, as shown in Fig 2, outlines 15 features (columns of the dataframe). Table. 1 provides the details of the information stored in each column along with the names. Furthermore, as can be seen from the extracted data (see Fig. 2), each row represents an entry for a user's data on a particular day. The user can be uniquely identified by the **userId** and has at least 56 days of data. The significance of the column **SicknessDate** is crucial for the purpose of our model development as a date entry in this column indicates that the user felt sick on that particular day and marked it. Therefore, this column is used as the **target variable** for the machine learning model development. In addition, the dataset includes 14 other columns with continuous numeric values and pandas objects that can be served or further manipulated to construct our **feature sets**. Our preliminary look into the data also reveals the presence of **NULL** values that we will explore and handle in Sec. 3.

a

	userId	dataDate	steps	activeMinutes	sedentaryMinutes	sleepEfficiency	minutesAsleep	sleepStartTime	wakeTime	restingHeartRate	atmPressure
0	31633	2019-03-05 00:00:00.000	4157	228	896	91	286	2019-03-04 22:20:30.000	2019-03-05 03:06:30.000	70.0	NaN
1	31633	2019-03-06 00:00:00.000	9674	277	710	96	409	2019-03-05 22:22:00.000	2019-03-06 05:11:00.000	71.0	NaN
2	31633	2019-03-07 00:00:00.000	10799	312	667	95	403	2019-03-06 22:27:00.000	2019-03-07 05:10:00.000	73.0	NaN
3	31633	2019-03-08 00:00:00.000	10811	369	743	90	293	2019-03-07 22:43:00.000	2019-03-08 03:36:00.000	74.0	1011.0
4	31633	2019-03-09 00:00:00.000	5819	264	785	97	379	2019-03-08 23:04:00.000	2019-03-09 05:23:00.000	72.0	1013.0
...	...	...	...	...	...	...	...	...	...	...	...
195	30776	2019-03-25 00:00:00.000	9298	356	669	94	386	2019-03-24 21:33:00.000	2019-03-25 03:59:00.000	80.0	1026.0
196	30776	2019-03-26 00:00:00.000	10855	413	648	84	318	2019-03-25 22:11:00.000	2019-03-26 03:29:00.000	79.0	1029.0

b

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22068 entries, 0 to 22067
Data columns (total 15 columns):
userId               22068 non-null int64
dataDate             22068 non-null object
steps                22068 non-null int64
activeMinutes        22068 non-null int64
sedentaryMinutes     22068 non-null int64
sleepEfficiency      22068 non-null int64
minutesAsleep        22068 non-null int64
sleepStartTime       22068 non-null object
wakeTime             22068 non-null object
restingHeartRate     22063 non-null float64
atmPressure          12206 non-null float64
humidity             12206 non-null float64
SicknessDate         1416 non-null object
energy               8344 non-null float64
feel                 8344 non-null float64
dtypes: float64(5), int64(6), object(4)
memory usage: 2.5+ MB
```

Figure 2: A quick look into the health data (*healthdata.csv*), as used in this project, reveals that each row is an entry for a user's data on a particular day. Note that part (a) only shows first 11 columns of the dataframe. Part (b) shows the name of the 15 columns along with the types of the data stored in the respective columns.

Table 1: Details of the health data from wearable devices utilized in this report

1.	userId	Unique identifier for each user
2.	dataDate	The date of the recorded data
3.	steps	Total number of steps recorded for this day
4.	activeMinutes	Total number of recorded minutes that the user was active (walking, running etc)
5.	sedentaryMinutes	Total number of recorded minutes that the user was inactive (sitting, laying etc)
6.	sleepEfficiency	Quantification of how efficient a sleep was as a percentage (0-100)
7.	minutesAsleep	Total number of minutes during that users main sleep for the day
8.	sleepStartTime	Time that the user started their main sleep for the day
9.	wakeTime	The result of sleepStartTime + minutesAsleep
10.	restingHeartRate	The average resting heart rate over the day in beats/minute
11.	atmPressure	The atmospheric pressure recorded at the users location on the given date (milibars)
12.	humidity	Air humidity recorded at the users location on the given date (percent)
13.	SicknessDate:	Date that a user logged a sickness. If there is a value in this column, it means they recorded a sickness for that day
14.	energy	The energy ratings that the user entered for the day
15.	feel	The feel ratings that the user entered for the day.

3 Exploratory Data Analysis

3.1 Conversion of Data Type

With a bird-eye view of the data in our hand, we will now perform a detailed exploration of the data using various statistical and visualization methods. As can be seen from Fig. 2, our dataframe contains four objects that represent date/time entries. This includes **dataDate**, **sleepStartTime**, **wakeTime** and **SicknessDate** (see Table 1 for further details). We need to convert them to pandas datetime format for further analysis. Now, **SicknessDate** contains information on whether a person felt sick or not sick on a particular day and a date entry in this column represents that the user felt sick. In contrast, whenever this column does not have any entry, we can assume that the user did not feel sick on that specific day. In order to further process our data, we converted each entry of **SicknessDate** column to either a value of 1 or 0, respectively representing a sick or a not sick entry and renamed this column as **Sick** and considered this as our target variable. Fig. 3a shows a few elements of this newly converted target variable. In addition, we also converted the entries of **dataDate**, **sleepStartTime** and **wakeTime** columns into pandas datetime and constructed a few more useful features. The details of these newly formed features are given in Sec. 3.2.

a

	userId	Sick
0	31633	0
1	31633	0
2	31633	0
3	31633	0
4	31633	1
5	31633	0
6	31633	0
7	31633	0
8	31633	0
9	31633	0

c

	userId	sleep_hour	sleephourmode	wake_hour	wakehourmode	penalty	sleepfactor
0	31633	22	22	3	5	-2	less
1	31633	22	22	5	5	0	perfect
2	31633	22	22	5	5	0	perfect
3	31633	22	22	3	5	-2	less
4	31633	23	22	5	5	-1	less
...	...	...	...	...	...	...	...
22063	582	21	21	5	4	1	excess
22064	582	22	21	6	4	1	excess
22065	582	22	21	8	4	3	excess
22066	582	22	21	4	4	-1	less
22067	582	22	21	5	4	0	perfect

b

	dataDate	dayofweek	weekend
0	2019-03-05	1	0
1	2019-03-06	2	0
2	2019-03-07	3	0
3	2019-03-08	4	0
4	2019-03-09	5	1
5	2019-03-10	6	1
6	2019-03-11	0	0
7	2019-03-12	1	0
8	2019-03-13	2	0
9	2019-03-14	3	0

d

	userId	minutesAsleep	bedtime	sleepLatency	sleepEfficiency
0	31633	286	314.285714	28.285714	91
1	31633	409	426.041667	17.041667	96
2	31633	403	424.210526	21.210526	95
3	31633	293	325.555556	32.555556	90
4	31633	379	390.721649	11.721649	97

Figure 3: (a) The newly formulated target variable **Sick** from **SicknessDate**. A value of 1 refers to an entry, which the user marked as a sick day (it belongs to the positive case in our model development). A value of 0 indicates that the user did not feel sick (belongs to the negative case). (b) View of **dayofweek** and **weekend** features, extracted from **dataDate**. (c)-(d) A look into a few more features (*e.g.*, penalty, sleep factor, bed time, sleep latency) related to a user's sleeping pattern.

3.2 Extraction of New Features

3.2.1 Day of the Week and Weekend/Weekday

To begin with, the **dataDate** feature can be useful to prepare two additional features, namely **dayofweek** and **weekend**, respectively capturing which day of the week the data was recorded and if the day falls in a weekday or weekend (see also Fig. 3b)

3.2.2 Penalty Due to Irregularity in Sleeping Pattern

On the other hand, the handlings of **sleepStartTime** and **wakeTime** was a bit more tricky. To elaborate, a **penalty** term for lack or excess amount of sleeping (delineates a user's deviation from his/her most common sleeping pattern) is developed and computed as follows:

1. Convert both **sleepStartTime** and **wakeTime** to pandas datetimes.
2. For each entry, extract the hour of the day from the datetime object and save them into two new features called **sleep_hour** (the hour of falling asleep) and **wake_hour** (the hour of waking-up).
3. Compute (a) the mode of the hour of falling asleep and (b) the mode of the hour of waking-up for each individual user (as the sleeping pattern can vary from user to user).
4. For each user, compute how much every entry in **sleep_hour** (a record of the hour of falling asleep) is deviated from its mode.
5. Also, for each user, compute how much every entry in **wake_hour** (a record of the hour of waking-up) is deviated from its mode.
6. A sum of step 4 and 5 provides an estimation of the **penalty** term.

For our computation we organized the penalty term into three categories, namely, 'perfect' sleep (**penalty** $= 0$), 'excess' sleep (**penalty** > 0) and 'less' sleep (**penalty** < 0). Alternatively, this

means that the **penalty** term is zero for any users able to follow their most common sleeping schedule on any particular day (thus we call it'perfect'). For a user who slept late and woke up early, receives a negative penalty score (we call it 'less'), whereas a user who slept early and woke up late, receives a positive penalty score (known as 'excess'). Finally, we use one-hot encoding to create three more features to represent whether an entry delineates **perfect**, **excess** or **less** amount of sleep. Fig. 3c illustrates a few elements of these features.

3.2.3 Total Time Spent in Bed and Sleep Latency

The total amount of time spent in bed plays an important role in determining sleep efficiency. The sleep efficiency is defined as the ratio of the total time asleep (**minutesAsleep**) and the total amount of time spent in bed (let us called it **bedtime**). Therefore, we calculate **bedtime** from the respective entries in column **minutesAsleep** and column **sleepEfficiency** as

$$bed\ time = \frac{minutes\ asleep}{sleep\ efficiency}.$$

Previous research on sleep behavior recommended that in order to have an improved sleeping habit, the total amount of time spent in bed (**bedtime**) should be equal to or close to the total amount of time asleep (also given in our dataset as **minutesAsleep**). The difference between these two is known as 'sleep latency', which we also computed and considered as another feature for our model development (**sleepLatency** = **bedtime-minutesAsleep**). According to the sleep specialists, it is important to minimize sleep latency to achieve a higher sleep efficiency. Fig. 3d shows a few elements of these features.

3.2.4 Splitting the Health Data into Two Dataframes (Sick = 1 and Sick = 0)

Finally, by carefully studying the health data for 'sick' and 'not sick' group, it appeared to be beneficial to split the entire health data into two separate dataframes; one containing data when

Sick = 1 (data representing the users feeling sick) and the other containing data when **Sick** = 0 (data representing the users feeling not sick). Throughout our analysis, we called them **Sick** = 1 and **Sick** = 0, respectively. Interestingly, exploring the data in this segmented way opens intriguing opportunity to systematically understand and visualize various statistical properties of the health data.

3.3 Removal of Redundant Data, Unrealistic Data and Imputation of Missing Values

3.3.1 Removal of Redundant Data

Throughout our process of feature generation, data conversion and extraction, we also dropped a few of the columns to avoid data redundancy. In two different stages of our exploratory data analysis, we dropped **userId**, **dataDate**, **sleepStartTime**, **wakeTime**, **sleephourmode**, **wakehourmode**, **sleep_hour**, **wake_hour**, **sleepfactor**.

3.3.2 Removing Unrealistic Values and Imputation of Missing Values

In the next step, our goal is to identify unrealistic values and remove them. The origin of these values can be linked with errors in data acquisition, malfunction of the sensory circuits or simply, a device at an idle stage that did not capture any data (*e.g.*, the user may not have been wearing the device). Interestingly, the amount of missing data is quite large (62.2%) for the features **energy** and **feel**. To tackle this issue in this exploratory data analysis, we focused on the unrealistic values of data from other features and handled them at first and then revisited the missing values in **energy** and **feel**. By analyzing the statistical properties of **Sick = 1** and **Sick = 0** dataframes, we can clean our data on the basis of a few criteria or universal truths. These criteria are:

1. The sum of total active time, total sedentary time and total time asleep in a given row can not exceed 1440 minutes (total minutes in a 24-hour period). In regard to our dataset, this

a

```
df_sick.isnull().sum()
```

userId	0
dataDate	0
steps	0
activeMinutes	0
sedentaryMinutes	0
sleepEfficiency	0
minutesAsleep	0
sleepStartTime	0
wakeTime	0
restingHeartRate	0
atmPressure	0
humidity	0
Sick	0
energy	385
feel	385
dayofweek	0
weekend	0
sleep_hour	0
wake_hour	0
sleephourmode	0
wakehourmode	0
sleep_penalty	0
wake_penalty	0
penalty	0
sleepfactor	0
excess	0
less	0
perfect	0
bedtime	0
sleepLatency	0

dtype: int64

b

```
df_notsick.isnull().sum()
```

userId	0
dataDate	0
steps	0
activeMinutes	0
sedentaryMinutes	0
sleepEfficiency	0
minutesAsleep	0
sleepStartTime	0
wakeTime	0
restingHeartRate	0
atmPressure	0
humidity	0
Sick	0
energy	4177
feel	4177
dayofweek	0
weekend	0
sleep_hour	0
wake_hour	0
sleephourmode	0
wakehourmode	0
sleep_penalty	0
wake_penalty	0
penalty	0
sleepfactor	0
excess	0
less	0
perfect	0
bedtime	0
sleepLatency	0

dtype: int64

Figure 4: The estimations of NULL values in different columns in **Sick = 1** and **Sick = 0** dataframes after cleaning the unphysical values.

means **activeMinutes** + **sedentaryMinutes** + **minutesAsleep** ≤ 1440.

2. The range of the atmospheric pressure (**atmPressure**) should be 640–1042 millibar. The upper and lower limits of the atmospheric pressure can be obtained from the respective atmospheric pressures of the cities with the lowest and highest altitudes in the world (respectively, Jericho and La Paz).

3. The maximum humidity of the air in percentage is 100%, meaning **humidity** ≤ 100.

4. It is unphysical to have resting heart rate (**restingHeartRate**) equals to zero. Therefore, **restingHeartRate** > 0. The most common range of resting heart rate is 50-70 bpm.

After removing the data with unphysical or erroneous values, we can check the amount of **NULL** in both sick and not sick dataframes. As shown in Fig 4a-b, both in the case of **df_sick** (**Sick** = 1)

and **df_notsick** (**Sick** = 0), all the columns except **energy** and **feel** have 100% non-NULL values. Since, both **energy** and **feel** can serve as potential indicators of sickness, we replaced these NULL values with the corresponding average energy and feel ratings for the respective sets of **Sick = 1** and **Sick = 0** dataframes.

3.4 Data Visualization

3.4.1 Visualization of Time Spent in Various Activities

Let us begin by analyzing the amount of total time users spent in three different activities, namely, walking/running (**activeMinutes**), sitting/laying (**sedentaryMinutes**) and sleeping (**minutesAsleep**) as shown in Fig. 5. As we see, these results are categorized in the condition of users feeling sick (**Sick = 1** $\rightarrow$ Fig. 5a) or not sick (**Sick = 0** $\rightarrow$ Fig. 5b). The respective statistical mean of each activity in each category is also calculated (see Table 2). Note that the outliers are also visible in Fig. 5 and determined as data points outside of the range of Q_1-1.5 IQR and Q_3+1.5 IQR. Here Q_1 and Q_3 are respectively the first and third quartile and IQR is the interquartile range. We can easily identify some of the data points that make less or no realistic sense. For example, in the case of **Sick = 1** data, some of data shows zero or nearly zero sedentary time, meaning that user spent the day either sleeping or walking/running, which does not make sense. Same is true in the case of data when **Sick = 0**. Similarly, a person should not have sedentary time very high (close to 21–22 hours!) as it means less amount of time for sleeping and walking/running activity and appears unrealistic, particularly for **Sick = 0** data. Therefore, we remove the outliers in sedentary minutes (**sedentaryMinutes**) for both **Sick = 1** and **Sick = 0** datasets. Importantly, the data depicting the duration of sleep needs to be considered carefully. For example, a person feeling not sick should not have close to zero minutes of sleeping time. Thus, this constitutes an outlier for **Sick = 0** data. However, it is entirely possible that a person who is feeling sick has not slept at all on a particular day. Therefore, we can keep those values, though they fall in the outlier regions,

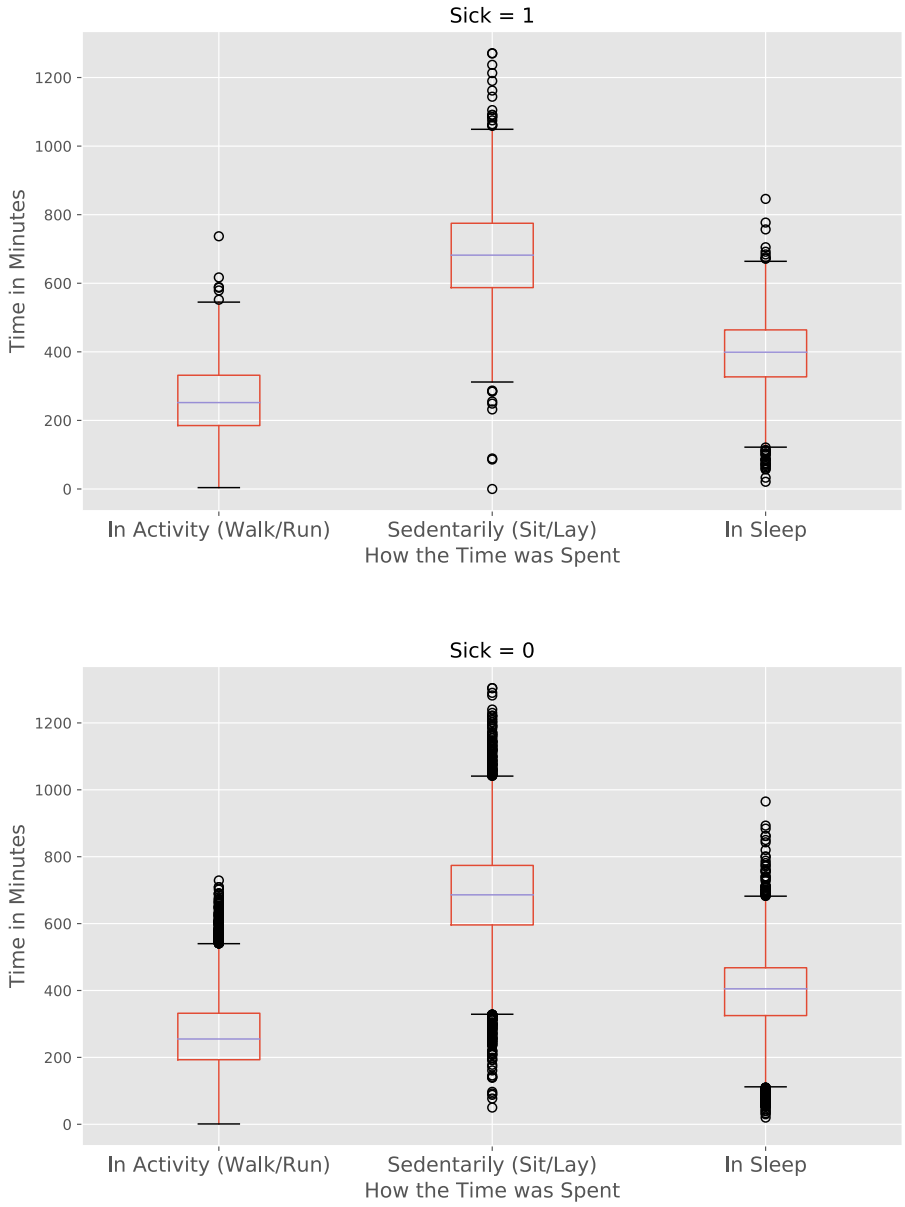

Figure 5: The boxplot representations of total time spent in three different activities for the case of **Sick = 1** (a) and **Sick = 0** (b). The outliers are defined as the data points outside of Q_1-1.5 IQR and Q_3+1.5 IQR.

Table 2: Distribution of times in daily activities for **Sick = 1** and **Sick = 0** datasets. The statistics of daily step count is also shown here

Features	**Sick = 1**	**Sick = 0**
Mean of total active minutes	260	269
Mean of total sedentary minutes	684	684
Mean of minutes asleep	392	394
Mean of daily steps	8712	8771

in regard to **Sick = 1** data due to the fact that both lack of sleep and too much of sleep can be ingredients for feeling sick. Consequently, we only remove the outliers in **minutesAsleep** for **Sick = 0** dataset. Next, in the case of the total amount of time spent in physical activity during a day, a value in between 550 - 750 minutes refers to 9 - 12.5 hours of activity. This might be true for athletes with **Sick = 0**. However, here we assumed it is highly unlikely that the dataset contains significant number of athletes as the users and we considered this higher range of active time as an outlier for dataset **Sick = 0**. Nevertheless, a person with excessive physical activity can feel sick and therefore, we keep those outliers for **Sick = 1** dataset.

To explore more, we also have raised the question: is there any dependency on how a user spends his time as a function of the day of the week(**dayofweek**) or weekend/weekdays (**weekend**)? Fig. 6 provides the insightful visualizations in this regard. The trends for **Sick = 1** dataset (top panel in Fig. 6) include:

- The daily portion of active minutes goes high and reaches the maximum on Thursday to Friday. Then it reduces and reaches its minimum on Sunday.
- Interestingly, the amount of sedentary minutes starts from a high value on Monday and gradually declines as the week reaches toward its end.
- The amount of minutes asleep follows the opposite trend of sedentary minutes as it increases from Monday $\rightarrow$ Sunday, where Sunday marks the maximum of daily minutes asleep.

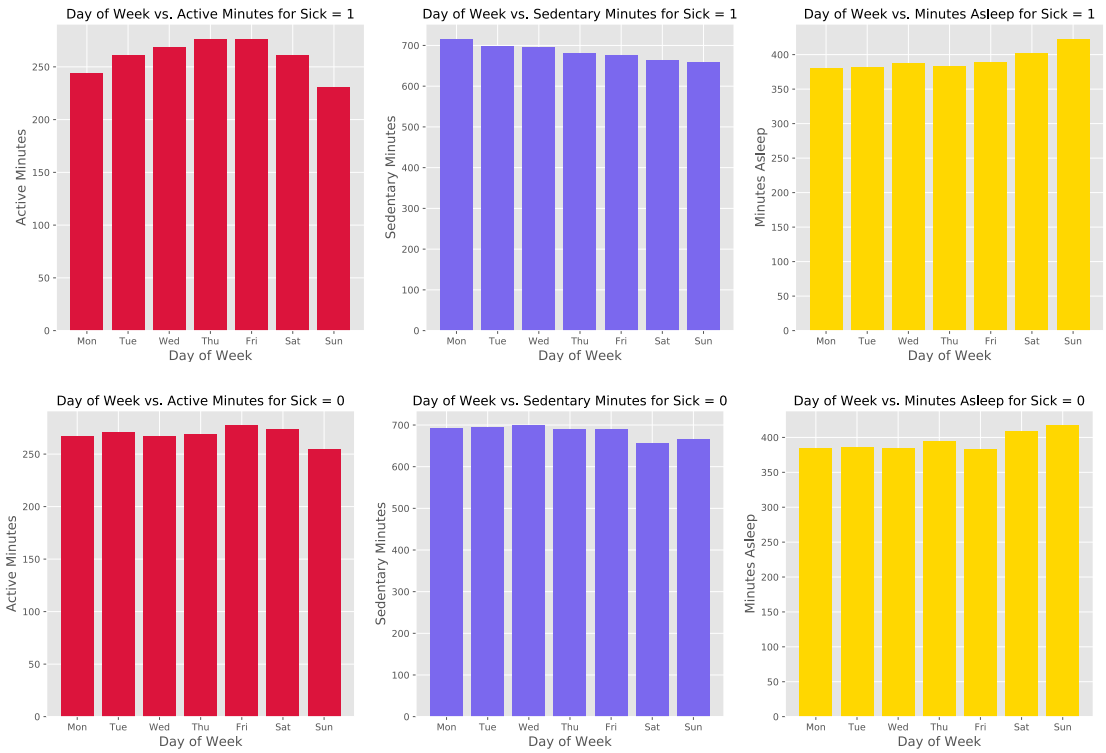

Figure 6: The average distribution of total time in a day distributed among three activities: walk/run, sit/lay and sleep as a function of the day of the week. Top panel shows the trends for **Sick = 1** and the bottom panel exhibits the trends for **Sick = 0**.

However, the similar trends for users who did not feel sick or **Sick = 0** (bottom panel in Fig. 6) are somewhat different with no monotonically rising pattern in minutes asleep or decreasing pattern in the amount of sedentary minutes. Also the amount of active time reaches its peak on Friday-Saturday.

3.4.2 Visualization of the Amount of Daily Steps

Fig. 7 visualizes the trends in number of steps taken by users feeling seek or users feeling not sick. As can seen from the top panel of Fig 7, it is hard to differentiate **Sick = 1** and **Sick = 0** in terms of the number of daily steps taken by a user and we removed the data outliers for both of the cases

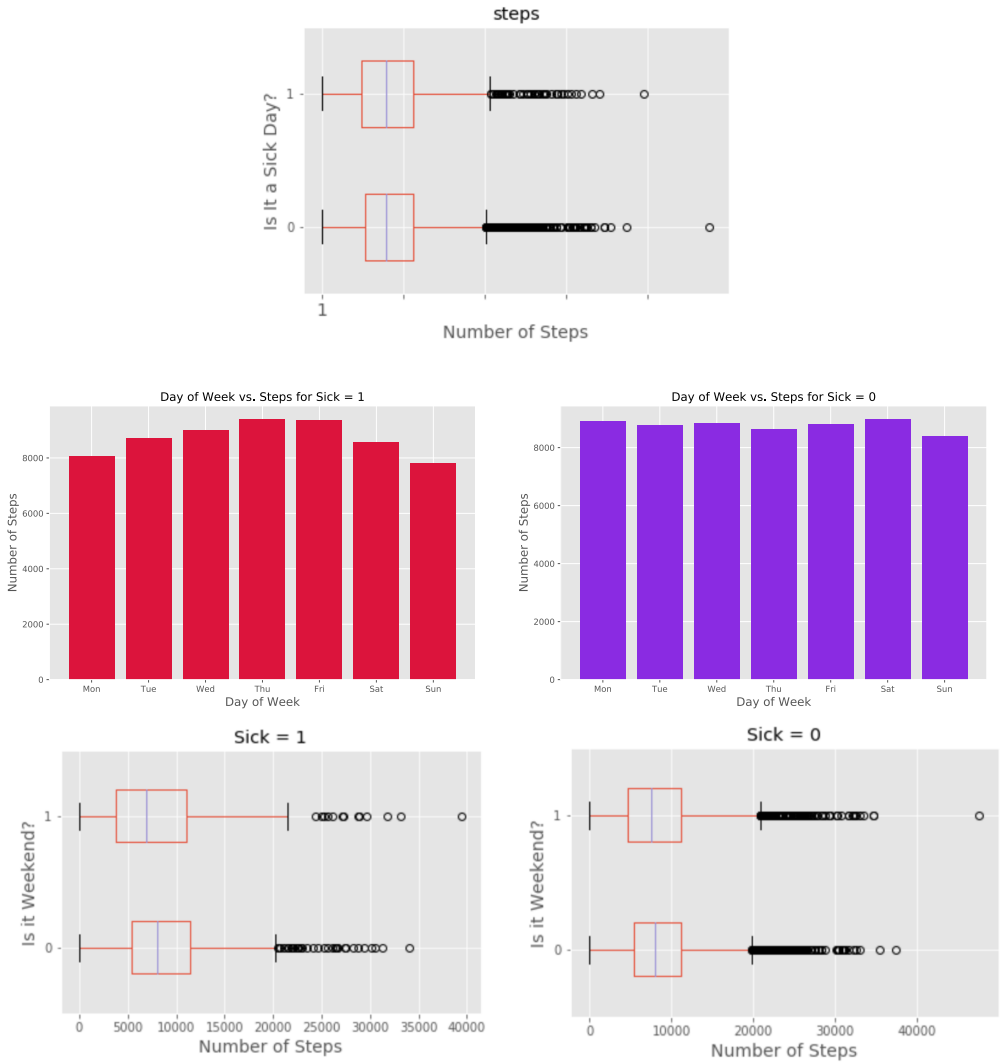

Figure 7: Top: the boxplot demonstrations of the statistics of number of steps as a function of **Sick = 1** and **Sick = 0**. Middle: The average distribution of number of steps taken calculated for different days of the week. Middle left panel shows the trends for **Sick = 1** and the middle right panel exhibits the trends for **Sick = 0**. Bottom: the boxplot demonstrations of the statistics of number of steps as a function of **weekend** for both **Sick = 1** (left) and **Sick = 0** (right) datasets.

considered here. However, we can extract a clear difference in the statistics of the number of steps taken by two of the classes considered in this study if we illustrate the trends as a function of the day of the week (see the middle panel in Fig. 7). In the case of users in class **Sick = 1**, the number of steps taken shows high variation and reaches its peak on Thursday-Friday, whereas the users in class **Sick = 0** exhibits less variation in the number of steps. The higher degree of variations in the number of steps taken by users feeling sick is also found to be consistent in the statistics as a function of weekend/weekday (see the bottom panel in Fig. 7).

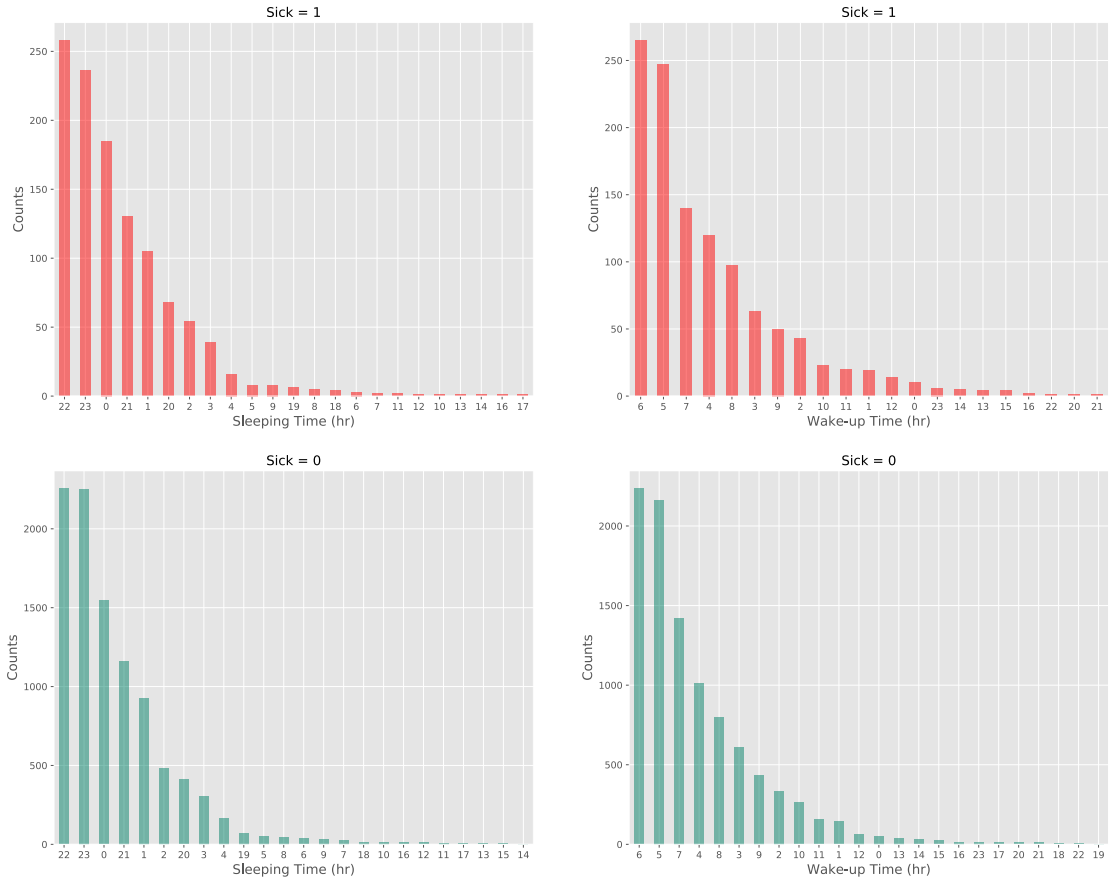

Figure 8: Top: counts of the hour of the day when the users with **Sick = 1** starts (left) and wakes up (right) from the main sleep of the day. Bottom: counts of the hour of the day when the users with **Sick = 0** starts (left) and wakes up (right) from the main sleep of the day.

3.4.3 Visualization of Sleep Related Data

In this section, we will visualize intriguing statistics of sleep related data to further comprehend the health data. The top panel of Fig. 8 exhibits the distributions of the hour of the day when the users with **Sick = 1** start sleeping (left) and wakes up (right) from the main sleep of the day. For the users in **Sick = 0** class, the same distributions are shown in the middle panel. It is surprising to see both of the classes follow a similar sleeping pattern and hard to differentiate at this format. However, if we consider our newly developed feature '**penalty**' (see Sec. 3.2 for details), we can start to see a few key differences in the sleeping pattern between these two classes. For example, the second largest group of users in **Sick = 1** class received a **penalty** = -1, meaning the users in this group have 1 hour of less sleep. Conversely, the the second largest group of users in **Sick = 0** class received a **penalty** = +1, indicating the users in this group have tend to have 1 hour of excess sleep. To enhance our comprehension of the sleeping pattern and how much a user deviates from his most common sleeping behaviour, we performed binning of the **penalty** feature into three categories, namely 'perfect', 'excess' and 'less' sleep (also see Sec. 3.2 for details). The result, as shown in the bottom panel of Fig. 8, is more intriguing. For example, the contribution from the constituents of class **Sick = 1** comes at a descending order of less > perfect > excess, whereas the contribution from the constituents of class **Sick = 0** comes at a descending order of excess > perfect > less. This visualization concludes that the users from **Sick = 1** class suffers from lack of sleep compared to the users from **Sick = 0** class.

We also explored the total amount of time spent in bed (**bedtime**) and sleep latency (**sleepLatency**) as a function of the day of the week and weekend/weekday. Fig. 10 summarizes our results. We found that with regard to the feature **bedtime** it is difficult to separate **Sick = 1** class from **Sick = 0** class. But both of these classes demonstrate higher amount of **bedtime** in the weekend compared to weekdays. However, we can see users in **Sick = 1** class tend to exhibit higher sleep latency in the early part of the week, whereas the users in **Sick = 0** class are more robust against

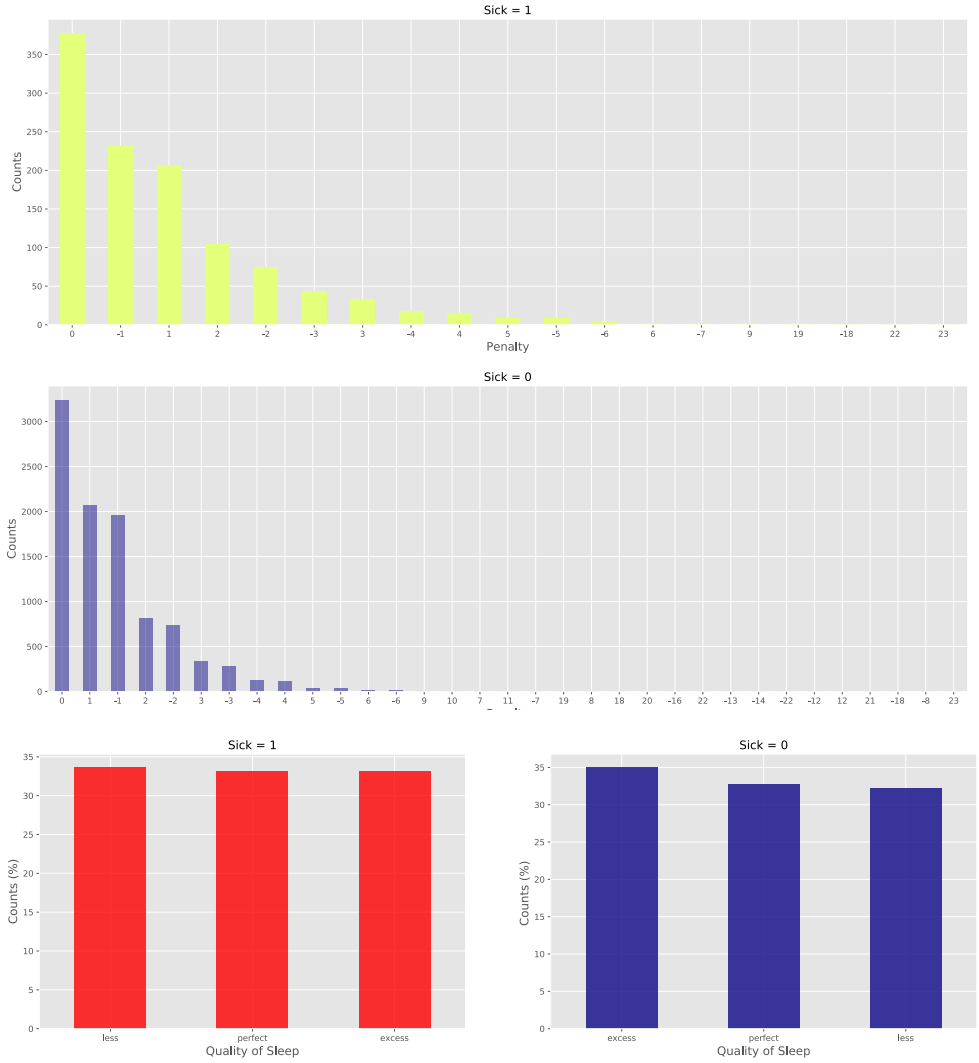

Figure 9: Top and Middle panels respectively show the distributions of **penalty** due to irregularity in the sleeping pattern for **Sick = 1** and **Sick = 0**. Bottom panel shows the binning of 'perfect','excess' and 'less' sleeping entries for **Sick = 1** (left) and **Sick = 0** (right).

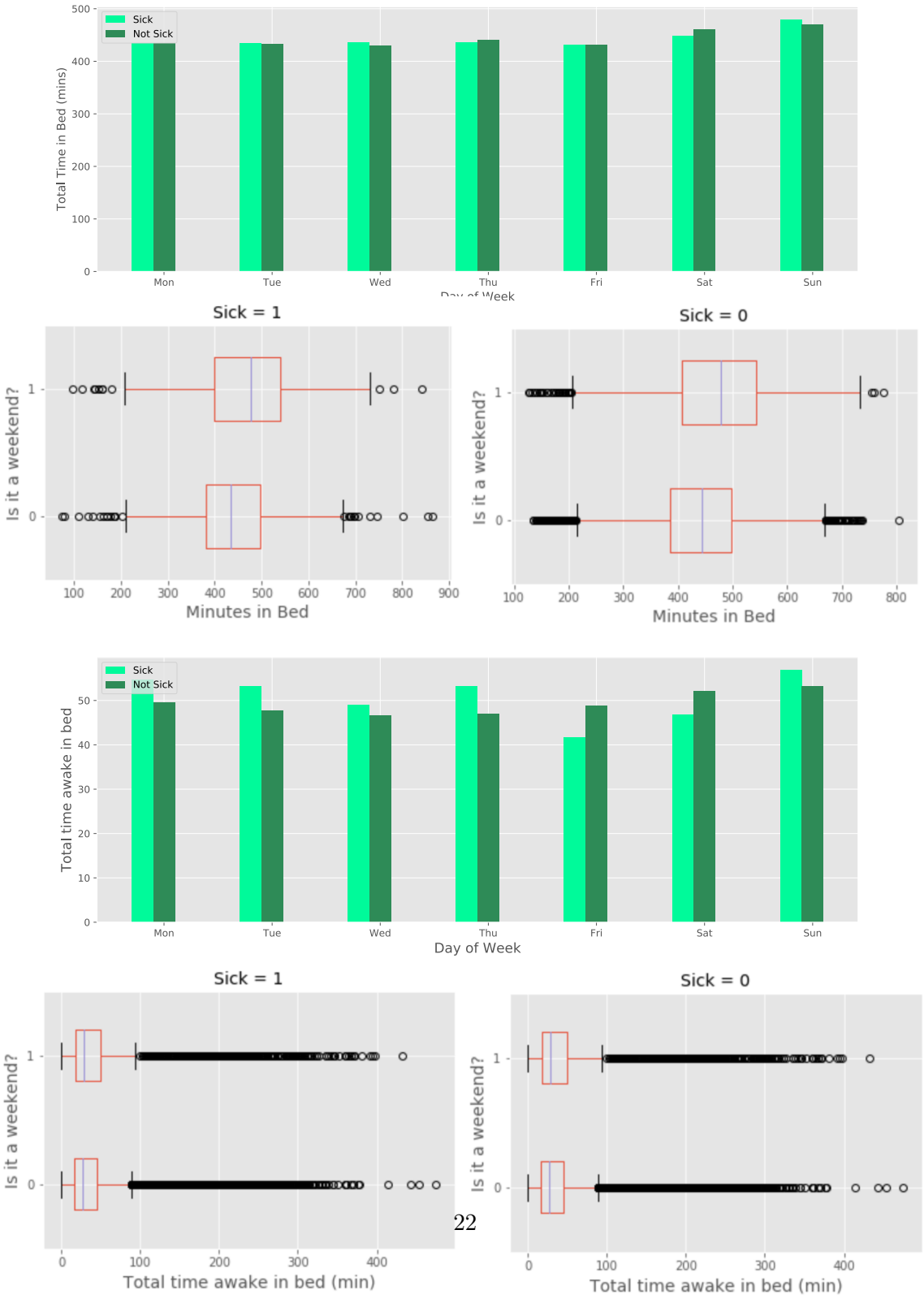

Figure 10: Visualizations of total amount of time spent in bed and sleep latency (total amount of time awake in bed) as a function of day of the week and weekend/weekday.

sleep latency.

3.4.4 Visualization of Resting Heart Rate, Humidity and Air Pressure

Our visualizations of the resting heart rate is shown in the top panel of Fig. 11. Most of the users tend to have a heart rate in the range of 50-70 bpm, which is acceptable and commonly reported in the literature as the normal range of heart rate. However, both the high and low heart rate are indicators of feeling discomfort and sick. A person consistently demonstrating a slow heart rate is believed to develop **bradycardia** and a person consistently demonstrating a fast heart rate is believed to develop **tachycardia**. Therefore, a person exhibiting resting heart rate below 50 bpm or above 80 bpm might be prone to these two medical conditions and feel sick. Consequently, the heart rate data in the low and high range (often detected as outliers) may be important to train our machine learning model for these conditions and we will not remove them from **Sick = 1**. However, we can remove the outliers in **Sick = 0** outside of 50-80 bpm, as general consensus is any heart rate outside of this range comes with discomfort and feeling of sickness.

In regard to the **humidity** as shown in the middle panel of Fig. 11, we can see from that the users feeling sick are often experienced higher humidity compared to **Sick = 0** class. Our data shows that the mean humidity of **Sick = 1** class ($\approx 69\%$) is higher than the mean humidity of **Sick = 0** class ($\approx 65\%$). Likewise, exposure to both high and low atmospheric pressure can cause discomfort and feeling of sickness. For instance, a low atmospheric pressure, due to high altitude of the user's location, makes it difficult to breath as the oxygen concentration of the air reduces. This expected trend is also visible in our computation of the statistics of air pressure for **Sick = 1** and **Sick = 0** classes (see the bottom panel of Fig. 11). We can see a higher number of users feeling sick are exposed to either high or low atmospheric pressure compared to the users in **Sick = 0** class.

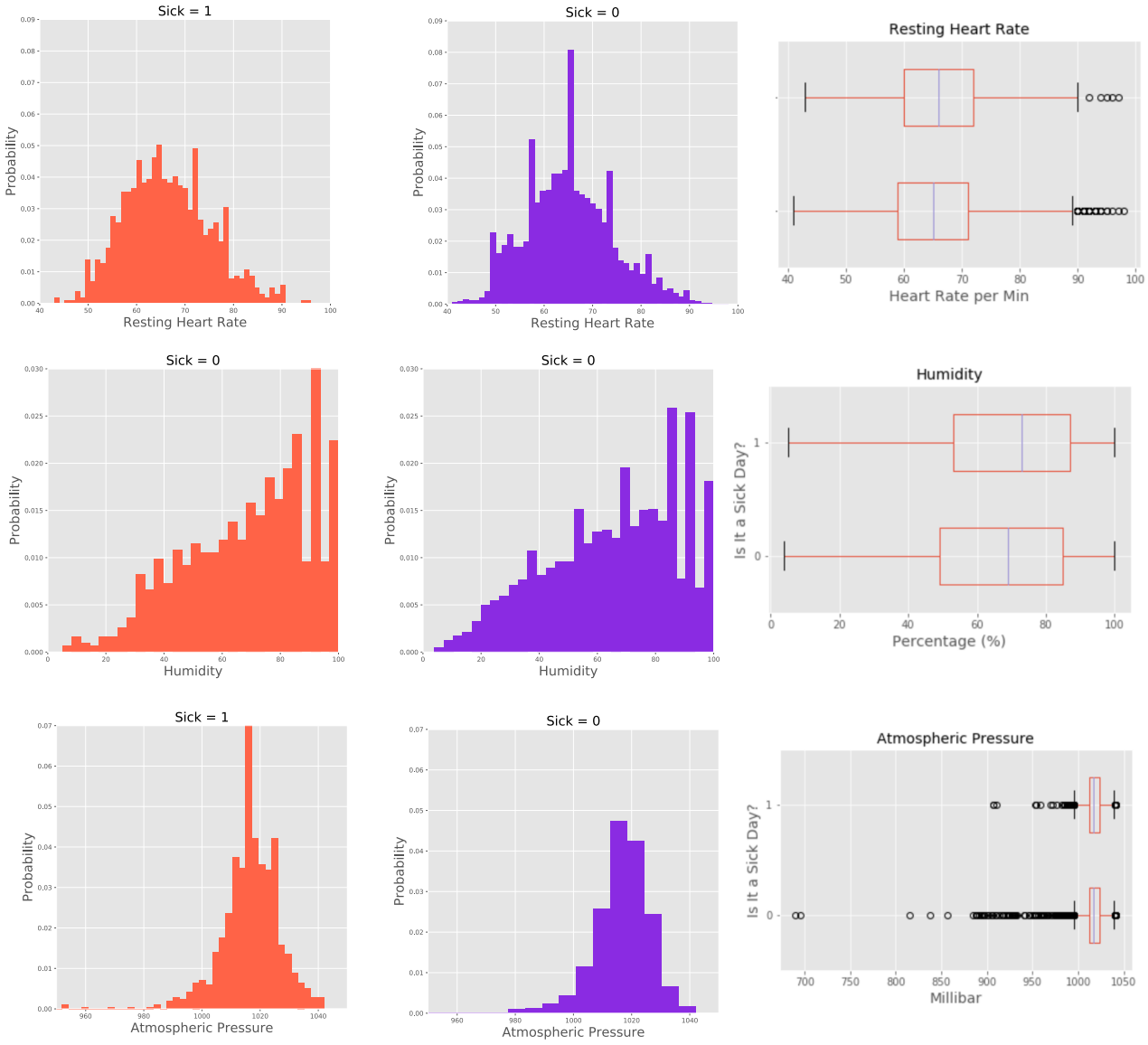

Figure 11: Visualizations of resting heart rate (top), humidity (middle) and atmospheric pressure (bottom) for **Sick = 1** and **Sick = 0** classes.

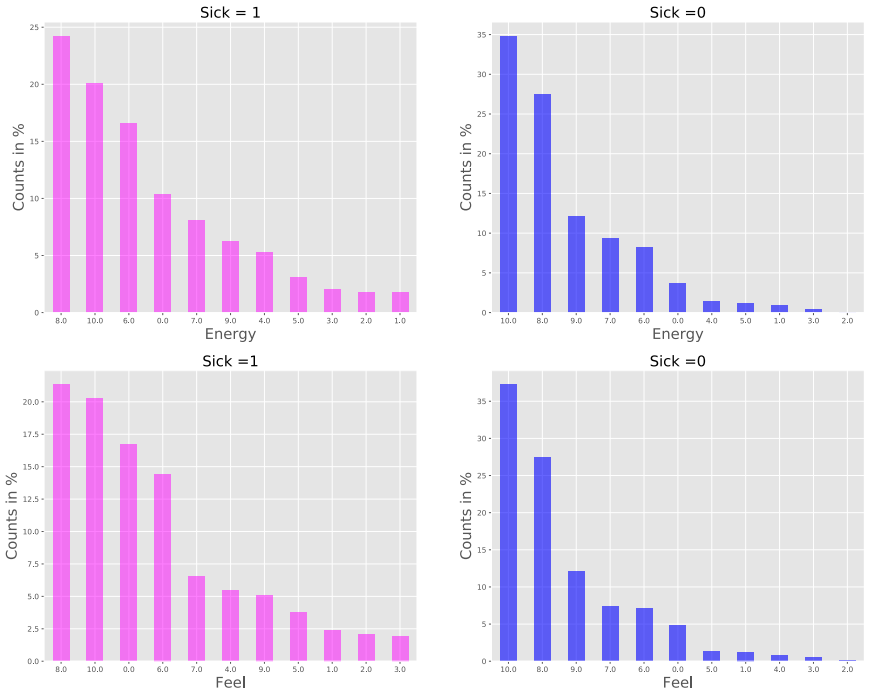

Figure 12: Visualizations of the **energy** (top) and **feel** (bottom) ratings for **Sick = 1** (left) and **Sick = 0** classes.

3.4.5 Visualization of Energy and Feel Ratings

Fig. 12 shows the distributions of **energy** (top) and **feel** (bottom) ratings for users in **Sick = 1** (left) and **Sick = 0** classes. The top four **energy** and **feel** ratings for **Sick = 1** class are:

- Energy: 8 (24%), 10 (20%), 0 (17%), 6 (11%).
- Feel: 8 (21%), 10 (20%), 6 (17%), 0 (14%).

Likewise, the top four **energy** and **feel** ratings for **Sick = 0** class are:

- Energy: 10 (35%), 8 (27.5%), 9 (12%), 7 (9%).

	count	mean	std	min	25%	50%	75%	max
steps	8198.0	8188.686631	3919.698252	38.000000	5279.000000	7592.500000	10631.250000	20615.000000
activeMinutes	8198.0	260.354233	93.497141	4.000000	193.000000	251.000000	321.000000	617.000000
sedentaryMinutes	8198.0	690.637229	119.889780	341.000000	610.000000	692.000000	772.750000	1031.000000
sleepEfficiency	8198.0	94.079410	3.300554	71.000000	92.000000	95.000000	97.000000	100.000000
minutesAsleep	8198.0	422.924982	84.446817	175.000000	370.000000	425.000000	478.000000	672.000000
restingHeartRate	8198.0	64.630886	7.584770	43.000000	59.000000	64.000000	70.000000	96.000000
atmPressure	8198.0	1017.329064	8.418228	993.000000	1012.000000	1018.000000	1023.000000	1041.000000
humidity	8198.0	65.504391	22.543178	10.000000	49.000000	69.000000	85.000000	100.000000
energy	8198.0	7.923459	1.947472	0.000000	8.000000	8.071849	9.000000	10.000000
feel	8198.0	7.854692	2.166194	0.000000	8.000000	8.075774	9.000000	10.000000
dayofweek	8198.0	2.937302	1.982902	0.000000	1.000000	3.000000	5.000000	6.000000
weekend	8198.0	0.268968	0.443450	0.000000	0.000000	0.000000	1.000000	1.000000
sleep_penalty	8198.0	-0.191998	1.693906	-12.000000	-1.000000	0.000000	0.000000	12.000000
wake_penalty	8198.0	0.338985	1.729724	-12.000000	0.000000	0.000000	1.000000	12.000000
penalty	8198.0	0.146987	1.876181	-18.000000	-1.000000	0.000000	1.000000	23.000000
excess	8198.0	0.358624	0.479626	0.000000	0.000000	0.000000	1.000000	1.000000
less	8198.0	0.302757	0.459479	0.000000	0.000000	0.000000	1.000000	1.000000
perfect	8198.0	0.338619	0.473269	0.000000	0.000000	0.000000	1.000000	1.000000
bedtime	8198.0	449.674950	89.459312	209.183673	393.684211	451.530772	508.602151	692.222222
sleepLatency	8198.0	26.749968	15.968875	0.000000	14.536082	23.935484	36.292420	93.473684

Figure 13: Statistical properties of the cleaned data that is used in the development of the machine learning model.

- Feel: 10 (38%), 8 (27.5%), 9 (12%), 7 (7%).

It is interesting to see that the dominant ratings by class **Sick = 1** are comprised of both low and high numbers, whereas the dominant ratings by class **Sick = 0** are comprised of high numbers. In addition, the top five **energy** and **feel** ratings given by class **Sick = 1** comprised of the same series of numbers and the top six **energy** and **feel** ratings given by class **Sick = 0** comprised of the same series of numbers.

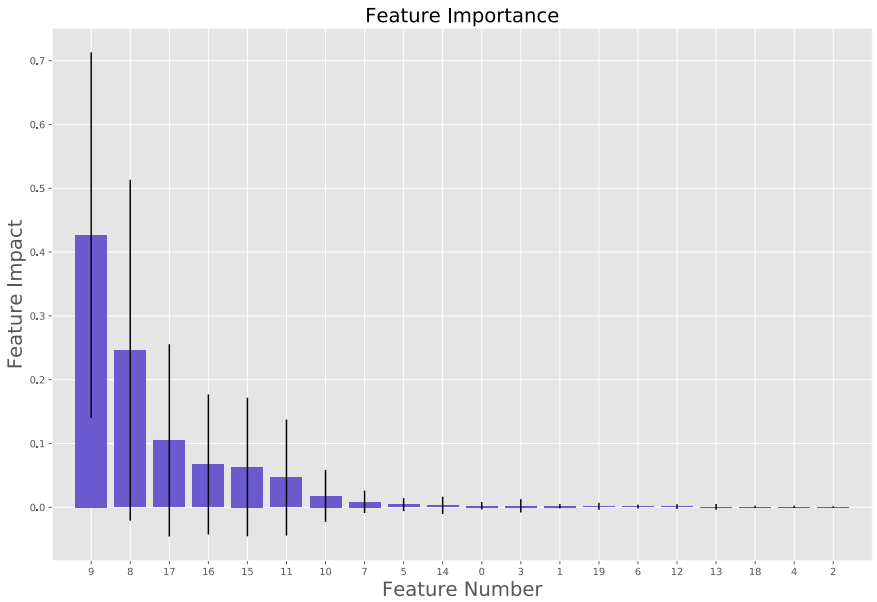

Figure 14: A tentative estimation of the importance of the features in the clean health data to separate **Sick = 1** and **Sick = 0** classes.

3.4.6 A Look into the Clean Data and its Features

Before we start our discussion on the machine learning model development, let us take a look into the cleaned version of the health data as shown in Fig. 13. This data is derived by following the footprint of our exploratory data analysis and ultimately recombining both **Sick = 1** and **Sick = 0** dataframes. Finally, to check the ability of this clean dataset to separate **Sick = 1** and **Sick = 0** classes, we fed a Random Forest (RF) classifier (not-optimized) from Scikit-Learn with our clean data. Fig. 14 exhibits that the features of the clean data has the potential to separate **Sick = 1** and **Sick = 0** classes and features **feel**, **energy**, **perfect**, **less**, **excess**, **weekend** have the most impact in the non-optimized RF classifier.

4 Machine Learning Model Development

4.1 Dealing with Imbalance Dataset

With the clean health data in our hand, we will now begin our discussion on how we can utilize the data and implement a machine learning model to predict user's sickness – the ultimate aim of this project. At this point of our discussion, it is important to note that the health dataset is highly imbalanced – only 11.2% of its entry belong to **Sick** = 1 class (see Table 3). In practice there exists a number of ways to deal with imbalance dataset while developing a machine learning model using the imbalance data. The generic approach is to perform random **up-sampling** of the minority class or random **down-sampling** of the majority class. Random up-sampling (also known as oversampling) can establish the balance in different class populations by repeating samples from the minority class. But in some cases it is prone to overfit a model. On the other hand, down-sampling randomly drops out samples from the majority class and consequently results in the loss of valuable information. However, there also exists a few sophisticated algorithms for up-sampling that do not simply duplicate minority class population, rather utilize interpolation techniques to create new samples near the existing samples of the minority class. The Synthetic Minority Oversampling Technique (SMOTE) is one of these algorithms, which we will use in this project to tackle imbalance in our data.

4.2 Splitting of Data into Training, Evaluation and Test Sets

Following the standard procedure, we also divided the data into three sets; training data (X_{train}, y_{train}), validation data (X_{val}, y_{val}) and test data (X_{test}, y_{test}). Here, (X, y) $\rightarrow$ (feature sets, target variable) and we considered a 80% : 10% : 10% splitting of the data into train, validation and test sets. Some of the key concepts to remember:

- Although a splitting of 80% : 10% : 10% is considered here, but we can also perform data

Table 3: Details of the data resampling and splittings into training, validation and test sets. Here X = features and y = target variable

Set	Dimension	Minority Class Population
X (before resampling)	(8198, 20)	11.2%
y (before resampling)	(8198,)	11.2%
X (after upsampling)	(14560, 20)	50%
y (after upsampling)	(14560,)	50%
X_{train} (80%)	(11648, 20)	50%
y_{train} (80%)	(11648,)	50%
X_{val} (10%)	(1456, 20)	50%
y_{val} (10%)	(1456,)	50%
X_{test} (10%)	(1456, 20)	50%
y_{test} (10%)	(1456,)	50%

splitting with a slightly different ratio (*e.g.*, 70% : 15% : 15%).

- As the name suggests, the training set is utilized to train the ML model, whereas the validation set provides an unbiased estimate of the model's performance while we develop the model or tune the hyperparameters. The performances on training data and validation data can be useful to tune the hyperparameters, accomplish the bias-variance tradeoff and decide which model to choose.

- The purpose of test set is to provide an unbiased estimate of the performance of the final model already fitted with the training data.

- It is highly imperative to note that the health dataset used in this project is imbalanced or skewed in character. The ratio of the number of observations in $\mathbf{Sick = 0}$ and $\mathbf{Sick = 1} \approx 7.93$ and only 11.2% of the total data belongs to $\mathbf{Sick = 1}$ class. Consequently, we utilized random upsampling and SMOTE technique to establish balance in the data.

- The accuracy of the model will not be a good indicator of how well the model can predict user's sickness due to the imbalance nature of the data. Consequently, we considered F1-score as the single number performance metric. Also, as the primary goal of this model is to

Table 4: Machine learning algorithms and the relevant hyperparameters

Classifier Model	Hyperparameters
Logistic Regression (LR)	C
Decision Tree (DT)	criterion, class_weight, max_depth, min_sample_split, max_features
Random Forest (RF)	criterion, min_samples_split, n_estimators, max_depth, max_features
Gradient Boosting (GB)	n_estimators, max_depth, learning_rate
Multi-Layer Perceptron (MLP)	hidden_layer_sizes, activation, learning_rate
Support Vector Machine (SVM)	kernel, C
K-Nearest Neighbors (KNN)	n_neighbors, weights

predict sickness, our intention will be to reduce the number of 'false negative' or to achieve a considerably high recall.

- When splitting the data into three different sets using Scikit-Learn library, we will make *stratify = True* to ensure consistent distributions of **Sick = 0** and **Sick = 1** among training, validation and test sets (which is not needed if the data is balanced).

- We will also use stratified K-folds (K = 5) cross-validator in the training phase to preserve the percentage of samples for each class.

4.3 Grid Search Approach to Find out the Best Model

In this project we have utilized Scikit-Learn machine learning library to implement the model predicting user's sickness. Following the exploratory data analysis, data cleaning, data resampling and train/validation/test data splits, we paired the stratified K-folds with Scikit-Learn's GridSearchCV to exhaustively tune variety of hyperparameters settings to find out the best set-up for each algorithm (see Fig. 1 for deatils). In this project, we have considered the following seven machine learning algorithms to fit the training data and perform the unbiased performance evaluation by validation dataset. The details of the hyperparameters used to tune each models are listed in Table 4. Finally, the best performing models from each machine learning algorithms were evaluated

Rank	Algorithm	Performance on Test Set	Hyperparameters
1	Gradient Booting Classifier	F1-Score = 0.97, Recall = 0.96	n_estimators = 300, max_depth = 9, learning_rate = 0.1
2	Random Forest Classifier	F1-Score = 0.97, Recall = 0.95	Criterion = entropy, min_samples_split = 10, n_estimators = 300, max_depth = 25, max_features = 8

Figure 15: Details of the top performance gradient boosting and random forest classifiers developed in this project (see also Fig. 16).

against the test data and the performance metrics were analyzed to find out the best working algorithm.

5 Results

Fig. 16 outlines the results the performance of the various machine learning models developed in this project. Throughout this model development, we considered F1-score as the single number performance metric. However, the application of this model in predicting a person's sickness also requires that the number of false negative must be minimized. Therefore, it is also important to ensure that the model delivers the best F1-score along with a high recall. Consequently, in order to assess the performance of our machine learning models and eventually pick up the best model from them, we sorted out the models in regard to two scorers, F1-score and recall. This is shown in Fig. 16. As can be seen, the logistic regression model performs poorly compared to the other algorithms. The performances of multi-layer preceptron, support vector machine and k-NN perform are comparable and all of them outperforms LR model. As we move to decision tree algorithm, we can see improvement in both F1-score and recall score. Finally, this study has found random forest and gradient boosting classifiers outcome the top two models for sickness prediction from wearable data. However, if we consider the time latency, we can easily conclude that the gradient boosting classifier is the top model, delivering a F1-score 0.97 and recall of 0.96.

Performance on Training Set								
Classifier	Scorer: F1- Score				Scorer: Recall Score			
	Mean F1-Score	Mean Recall	Mean Precision	Mean Accuracy	Mean F1-Score	Mean Recall	Mean Precision	Mean Accuracy
LR	0.737	0.696	0.784	0.752	0.737	0.696	0.784	0.752
DT	0.926	0.918	0.933	0.926	0.924	0.915	0.933	0.924
RF	0.956	0.936	0.976	0.956	0.953	0.938	0.969	0.954
GB	0.960	0.937	0.985	0.961	0.960	0.937	0.984	0.961
MLP	0.910	0.913	0.907	0.910	0.904	0.911	0.898	0.904
SVM	0.907	0.899	0.915	0.908	0.907	0.899	0.915	0.908
KNN	0.896	0.976	0.828	0.886	0.891	0.977	0.819	0.880

Performance on Validation Set										
Classifier	Scorer: F1- Score					Scorer: Recall Score				
	Mean F1-Score	Mean Recall	Mean Precision	Mean Accuracy	Latency (ms)	Mean F1-Score	Mean Recall	Mean Precision	Mean Accuracy	Latency (ms)
LR	0.735	0.709	0.762	0. 744	1.0	0.735	0.709	0.762	0.744	1.2
DT	0.913	0.912	0.915	0.913	0.5	0.932	0.924	0.94	0.933	0.7
RF	0.958	0.940	0.977	0.959	105.3	0.957	0.940	0.974	0.957	107.1
GB	**0.962**	**0.944**	**0.98**	**0.962**	**27.6**	**0.959**	**0.944**	**0.974**	**0.959**	**29**
MLP	0.909	0.924	0.895	0.908	2.3	0.906	0.915	0.898	0.905	2.5
SVM	0.913	0.905	0.920	0.913	249.0	0.913	0.905	0.92	0.913	233.8
KNN	0.907	0.979	0.845	0.9	209.3	0.902	0.981	0.835	0.894	315.5

Performance on Test Set										
Classifier	Scorer: F1- Score					Scorer: Recall Score				
	Mean F1-Score	Mean Recall	Mean Precision	Mean Accuracy	Latency (ms)	Mean F1-Score	Mean Recall	Mean Precision	Mean Accuracy	Latency (ms)
LR	0.747	0.707	0.791	0. 760	1.2	0.747	0.707	0.791	0.760	0.6
DT	0.917	0.904	0.931	0.918	0.5	0.936	0.934	0.938	0.936	0.7
RF	**0.970**	**0.952**	**0.989**	**0.970**	**102.4**	0.968	0.956	0.980	0.968	103.6
GB	0.968	0.956	0.980	0.968	27.6	**0.972**	**0.960**	**0.985**	**0.973**	**28.4**
MLP	0.924	0.933	0.916	0.924	2.3	0.909	0.908	0.910	0.909	2.3
SVM	0.911	0.911	0.912	0.911	238.3	0.911	0.911	0.912	0.910	230.9
KNN	0.907	0.979	0.845	0.9	209.3	0.892	0.973	0.824	0.880	206.7

Figure 16: Tables containing the performance of different machine learning models developed in this work; top: performance on training set, middle: performance on the validation set and bottom: performance on test set. The results are computed with two scorer: best F1-score (left) and best recall score.