Predicting Injuries Among Combat Soldiers

Spring 2024

Tamir Offen - 211621479

Advisor: Barak Gahtan

Introduction

Injuries among combat soldiers are a significant concern for the military, impacting not only the health and well being of the soldiers but also the overall operational readiness and effectiveness of the military units. Predicting and preventing these injuries is crucial to maintaining high performance in combat units and reducing long term healthcare costs associated with military service. The ability to anticipate injuries allows for better resource allocation, training modifications, and preemptive medical interventions, ultimately leading to more efficient missions and enhanced soldier morale.

This project aims to forecast injuries and identify critical factors influencing soldier health by utilizing data collected from wearables worn by the Golani troop over a six month period. In collaboration with the physical therapy department of Haifa University and the Technion VISTA Lab and funded by the IDF, this project applies deep learning models to analyze large scale, real world data.

Related Work

Wearable Sensors in a Military Settings:

The use of wearable sensors for monitoring and predicting physical activity in military personnel has gotten a lot attention. One notable study by Papadakis et al. (2023) employed body fixed sensors (BFS) combined with ML techniques to recognize physical activities such as walking, running, and jumping in military soldiers. The research demonstrated the feasibility of using BFS to accurately predict activities and classify soldiers based on factors like gender and fitness level. Random Forest classifiers were identified as most effective, getting an accuracy of 92.9% on the validation set. This paper demonstrates the potential of wearable devices and a trained model in supporting military operations by providing insights into soldier performance and fatigue to the commanders.

Garmin Watches:

The BFS used in this project were Garmin watches. Garmin watches collect data about the wearer's physical activities, like steps, heart rate, and sleep. Garmin offers a Health API that allows researchers to effectively use the data.

Method

The initial phase of our project involved processing and preparing the data collected from the Garmin watches worn by the soldiers, ensuring it was properly formatted and optimized for use in a neural network. The data collected from the watches that we used were steps, heart rate, and sleep. From the data collected from the watches, we processed them into grids.

The code for the project can be found here

https://github.com/TamirOffen/Predicting_Injuries_in_Soldiers

Steps

Our objective is to organize the steps dataset into weekly grids for each soldier with 15 minute time intervals.

A notebook showing the steps grids creation is in steps_from_epochs.ipynb.

The steps data can be in found in *epochs.csv*. There are a total of 2,119,020 rows and 15 columns. Each row corresponds to a 15 minute (900 seconds) recording for a specific soldier at a certain start time. The following example shows the first 10 rows with relevant columns:

	userId	activityType	steps	distanceInMeters	durationInSeconds	activeTimeInSeconds	startTimeInSeconds	startTimeOffsetInSeconds
0	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1506	125.73	900.0	860	1650484963	-18000
1	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1174	207.00	900.0	584	1650485863	-18000
2	e31d5fa7-7a63-43a6-973a-f2169c0661f7	SEDENTARY	0	0.00	900.0	143	1650485863	-18000
3	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1363	222.45	900.0	188	1650486763	-18000
4	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1755	461.11	900.0	269	1650487663	-18000
5	e31d5fa7-7a63-43a6-973a-f2169c0661f7	SEDENTARY	0	0.00	900.0	133	1650487663	-18000
6	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1570	143.91	900.0	862	1650488563	-18000
7	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1090	74.24	900.0	961	1650489463	-18000
8	e31d5fa7-7a63-43a6-973a-f2169c0661f7	SEDENTARY	0	0.00	900.0	834	1650489463	-18000
9	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1073	459.05	900.0	525	1650490363	-18000

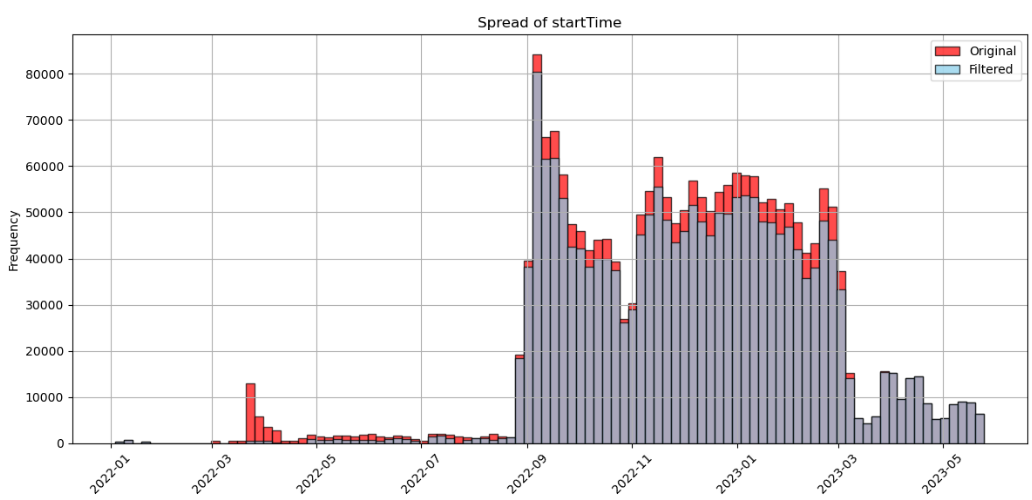
- *userId* – The soldier's ID. There are 213 unique IDs in this dataset.
- *activityType* – The activity type identified by the Garmin watch for the timespan. Can be either sedentary, walking, generic, running, or unmonitored.

- *startTimeInSeconds* – Start time of the activity, as a Unix timestamp (number of seconds since January 1st, 1970, UTC +0).
- *startTimeOffsetInSeconds* – The offset in seconds to add to startTimeInSeconds to derive the "local" time of the device that recorded the data.

To get the local time of each activity, we need to add *startTimeInSeconds* and *startTimeOffsetInSeconds*, and convert it to a UTC format. By adding the offset to the start time, we take into account the time zone in which the activity was recorded in. In Israel, there are two time zones throughout the year, UTC +2 (+7200 seconds) during winter and UTC +3 (+10800 seconds) during summer. It is important to use the local time to be consistent across all datasets. From the local time, we can calculate the week number of each activity, keeping into account that in Israel the week starts on a Sunday.
The following shows the first 10 rows of the dataset with local time and week number.

	userId	activityType	steps	distanceInMeters	durationInSeconds	activeTimeInSeconds	startTimeLocal	WeekNumber
0	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1506	125.73	900.0	860	2022-04-20 15:02:43	15
1	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1174	207.00	900.0	584	2022-04-20 15:17:43	15
2	e31d5fa7-7a63-43a6-973a-f2169c0661f7	SEDENTARY	0	0.00	900.0	143	2022-04-20 15:17:43	15
3	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1363	222.45	900.0	188	2022-04-20 15:32:43	15
4	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1755	461.11	900.0	269	2022-04-20 15:47:43	15
5	e31d5fa7-7a63-43a6-973a-f2169c0661f7	SEDENTARY	0	0.00	900.0	133	2022-04-20 15:47:43	15
6	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1570	143.91	900.0	862	2022-04-20 16:02:43	15
7	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1090	74.24	900.0	961	2022-04-20 16:17:43	15
8	e31d5fa7-7a63-43a6-973a-f2169c0661f7	SEDENTARY	0	0.00	900.0	834	2022-04-20 16:17:43	15
9	e31d5fa7-7a63-43a6-973a-f2169c0661f7	WALKING	1073	459.05	900.0	525	2022-04-20 16:32:43	15

About 10 percent of the data has offsets that are not +7200 or +10800 seconds. We want to remove this faulty data because the offset was not set properly, and the correct local time cannot be reliably calculated. The following graph shows the distribution of local times, both original data and the data filtered to have only correct offsets.



A detail to note about the dataset is that the startTimeInSeconds can be the same multiple times. This happens when the user has activities of different intensities during the same 15 minute period. This can be seen in the examples above. We chose to handle this by grouping by start time and adding the steps together for each 15 minute period. The following is the first 10 rows of the dataset after group by, resulting in 7 rows.

	userId	startTimeLocal	steps	distanceInMeters	activeTimeInSeconds
0	e31d5fa7-7a63-43a6-973a-f2169c0661f7	2022-04-20 15:02:43	1506	125.73	860
1	e31d5fa7-7a63-43a6-973a-f2169c0661f7	2022-04-20 15:17:43	1174	207.00	727
2	e31d5fa7-7a63-43a6-973a-f2169c0661f7	2022-04-20 15:32:43	1363	222.45	188
3	e31d5fa7-7a63-43a6-973a-f2169c0661f7	2022-04-20 15:47:43	1755	461.11	402
4	e31d5fa7-7a63-43a6-973a-f2169c0661f7	2022-04-20 16:02:43	1570	143.91	862
5	e31d5fa7-7a63-43a6-973a-f2169c0661f7	2022-04-20 16:17:43	1090	74.24	1795
6	e31d5fa7-7a63-43a6-973a-f2169c0661f7	2022-04-20 16:32:43	1073	459.05	525

We want to derive two more features from the datasets: speed and is_running. Speed calculation in the steps grid is the average speed in km/h of the soldier in the 15 minute interval when he is walking or running, which means only looking at activityTypes that are walking or running. We do this because the soldier is often resting in his 15 minute interval, so it wouldn't make sense to include that in his speed. The walking/running threshold is 7.5 km/h The following is an example:

	startTimeLocal	steps	userId	activeTimeInSeconds	distanceInMeters	speed	is_running
50	2022-09-05 02:45:00	147	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	115.04	1.380480	False
51	2022-09-05 03:00:00	366	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	298.18	1.987867	False
52	2022-09-05 03:15:00	30	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	23.48	0.704400	False
53	2022-09-05 03:30:00	290	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	226.95	1.945286	False
54	2022-09-05 03:45:00	767	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	790.42	4.742520	False
55	2022-09-05 04:00:00	111	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	86.87	1.303050	False
56	2022-09-05 04:15:00	415	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	461.74	9.234800	• True
57	2022-09-05 04:30:00	1997	29f6135a-5e81-4cde-92d6-4cfe5ad63547	840	2294.19	9.832243	• True
58	2022-09-05 04:45:00	163	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	127.56	1.275600	False
59	2022-09-05 05:00:00	129	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	100.96	1.009600	False

The following is the structure of the steps grids:

Level	Description
`users_weekly_epoch`	The main dictionary containing all users' data.
`userID`	A unique identifier for each soldier.
`week_number`	A key within each `userID` dictionary representing a specific week.
`pd.DataFrame`	A dataframe filled with the relevant data for that user and week. Split by 15 min intervals.

Each dataframe contains the following data:

- `startTimeLocal` – local time of user, example: 2022-08-28 19:15:00
- `steps` – number of steps in the 15 minute interval
- `userId` – soldier's ID
- `activeTimeInSeconds` – number of seconds the user was active in the 15 minute interval. i.e. not sedentary
- `distanceInMeters` – distance travelled by soldier in the 15 minute interval
- `speed` – average speed of soldier in 15 minute interval in km/h, of when he was walking/running.
- `is_running` – True if soldier was running (speed >= 7.5 km/h) in the 15 minute interval, False o.w.

Explanation:

- *users_weekly_epoch* is a dictionary keyed by the userId.
- *users_weekly_epoch*[*soldierID*] is a dictionary keyed by week number.
- *users_weekly_epoch*[*soldierID*][*week_num*] is a dataframe containing the corresponding steps grid for the soldier in week_num.

Example:

```
users_weekly_epoch['29f6135a-5e81-4cde-92d6-4cfe5ad63547'][35]
```

Executed at 2024.09.29 13:59:11 in 11ms

1-10 615 rows × 7 columns

	startTimeLocal	steps	userId	activeTimeInSeconds	distanceInMeters	speed	is_running
0	2022-09-04 14:15:00	86	29f6135a-5e81-4cde-92d6-4cfe5ad63547	480	70.51	3.213114	False
1	2022-09-04 14:30:00	370	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	289.56	1.737360	False
2	2022-09-04 14:45:00	185	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	144.78	1.447800	False
3	2022-09-04 15:00:00	363	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	284.08	2.434971	False
4	2022-09-04 15:15:00	150	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	117.39	1.408680	False
5	2022-09-04 15:30:00	773	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	604.95	4.537125	False
6	2022-09-04 15:45:00	331	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	259.04	1.726933	False
7	2022-09-04 16:00:00	363	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	284.09	3.409080	False
8	2022-09-04 16:15:00	106	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	82.95	0.995400	False
9	2022-09-04 16:30:00	140	29f6135a-5e81-4cde-92d6-4cfe5ad63547	900	109.57	1.643550	False

Heart Rate

Our objective is to organize the heart rate datasets into weekly grids for each soldier with 15 second time intervals.

A notebook showing the steps grids creation is in create_heartrate_grid.ipynb.

The heart rate dataset is made up of two datasets. The first is called *heart_rate_daily.csv* and there are 55,575,886 rows. The following is an example 5 rows from the dataset.

	userId	dailiessummaryId	timeOffsetHeartRateSamples	pulse
0	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6313c050-15180-6	33615	83
1	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6313c050-15180-6	33630	83
2	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6313c050-15180-6	33645	83
3	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6313c050-15180-6	33660	83
4	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6313c050-15180-6	33675	83

- *userId* is the ID of the soldier
- *dailiessummaryId* is an ID used in another dataset (*dailies_summary.csv*) to access information like calendar date.
- *timeOffsetHeartRateSamples* is the offset in seconds.

The second dataset is called *dailies_summary.csv* and it has 24,815 rows and many categories, but we only ended up using 3. The following is an example 5 rows from the dataset.

	userId	summaryId	calendarDate
0	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6313c050-15180-6	2022-09-04
1	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6313c050-8430-6	2022-09-04
2	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-631511d0-15180-6	2022-09-05
3	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-63166350-15180-6	2022-09-06
4	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6317b4d0-3cf0-6	2022-09-07

- *summaryId* corresponds with *dailiessummaryId* from the heart_rate_daily dataset.

- *calendarDate* is the local date, so no need to account for time zones.
- A very small number of entries for *calendarDate* are null, so we decided to just remove these rows.

To get the dataset that we will use when creating our grids, we will add a *datetime* and *WeekNumber* columns to *heart_rate_daily.csv* by joining it with *dailies_summary.csv*.

- *datetime* = calendarDate(dailies_summary.csv) + timeOffsetHeartRateSamples(heart_rate_daily.csv)
- *WeekNumber* = calendar week number, assuming weeks start on a Sunday.

Example:

	userId	dailiessummaryId	pulse	datetime	WeekNumber
0	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6313c050-15180-6	83	2022-09-04 09:20:15	35
1	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6313c050-15180-6	83	2022-09-04 09:20:30	35
2	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6313c050-15180-6	83	2022-09-04 09:20:45	35
3	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6313c050-15180-6	83	2022-09-04 09:21:00	35
4	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-6313c050-15180-6	83	2022-09-04 09:21:15	35

The following is the structure of the heart rate grids:

```
users_weekly_hr = {
    userID: {
        week_number: pd.DataFrame(userId, dailiessummaryId, pulse, datetime, WeekNumber)
    }
}
```

Explanation:

- *users_weekly_hr* is a dictionary keyed by userId.
- *users_weekly_hr*[*soldierID*] is a dictionary keyed by week number.
- *users_weekly_hr* [*soldierID*][*week_num*] is a dataframe containing the corresponding heart rate grid for the soldier in week_num.

Example:

```
users_weekly_hr['29f6135a-5e81-4cde-92d6-4cfe5ad63547'][35]
```

Executed at 2024.10.01 14:00:40 in 21ms

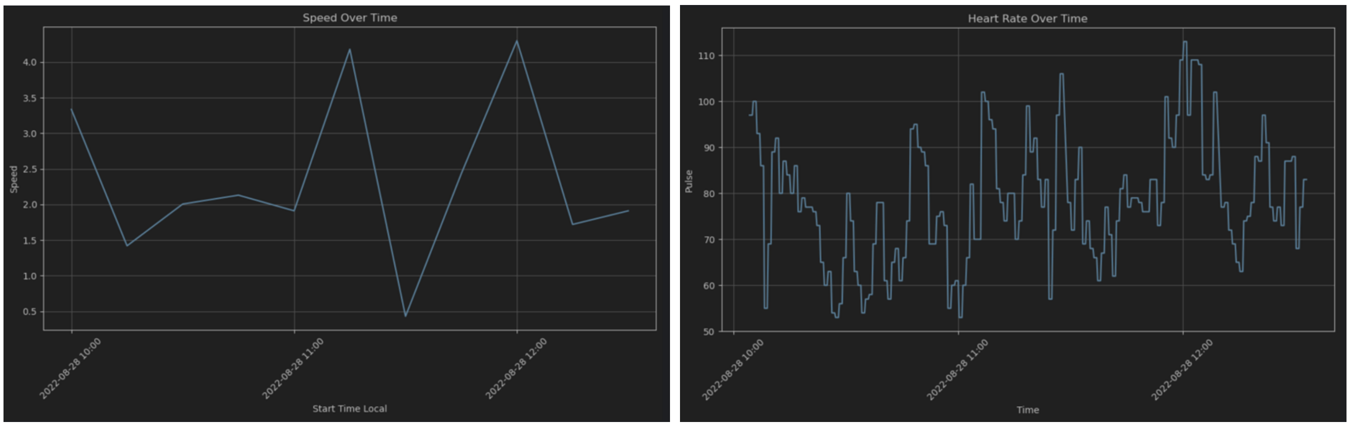
1-10 23464 rows × 5 columns

	userId	dailiessummaryId	pulse	datetime	WeekNumber
0	29f6135a-5e81-4cde-92d6-4cfe5ad63547	x48cc078-6313c050-10af4-6	82	2022-09-04 14:26:15	35
1	29f6135a-5e81-4cde-92d6-4cfe5ad63547	x48cc078-6313c050-10af4-6	82	2022-09-04 14:26:30	35
2	29f6135a-5e81-4cde-92d6-4cfe5ad63547	x48cc078-6313c050-10af4-6	82	2022-09-04 14:26:45	35
3	29f6135a-5e81-4cde-92d6-4cfe5ad63547	x48cc078-6313c050-10af4-6	82	2022-09-04 14:27:00	35
4	29f6135a-5e81-4cde-92d6-4cfe5ad63547	x48cc078-6313c050-10af4-6	82	2022-09-04 14:27:15	35
5	29f6135a-5e81-4cde-92d6-4cfe5ad63547	x48cc078-6313c050-10af4-6	82	2022-09-04 14:27:30	35
6	29f6135a-5e81-4cde-92d6-4cfe5ad63547	x48cc078-6313c050-10af4-6	82	2022-09-04 14:27:45	35
7	29f6135a-5e81-4cde-92d6-4cfe5ad63547	x48cc078-6313c050-10af4-6	82	2022-09-04 14:28:00	35
8	29f6135a-5e81-4cde-92d6-4cfe5ad63547	x48cc078-6313c050-10af4-6	78	2022-09-04 14:28:15	35
9	29f6135a-5e81-4cde-92d6-4cfe5ad63547	x48cc078-6313c050-10af4-6	78	2022-09-04 14:28:30	35

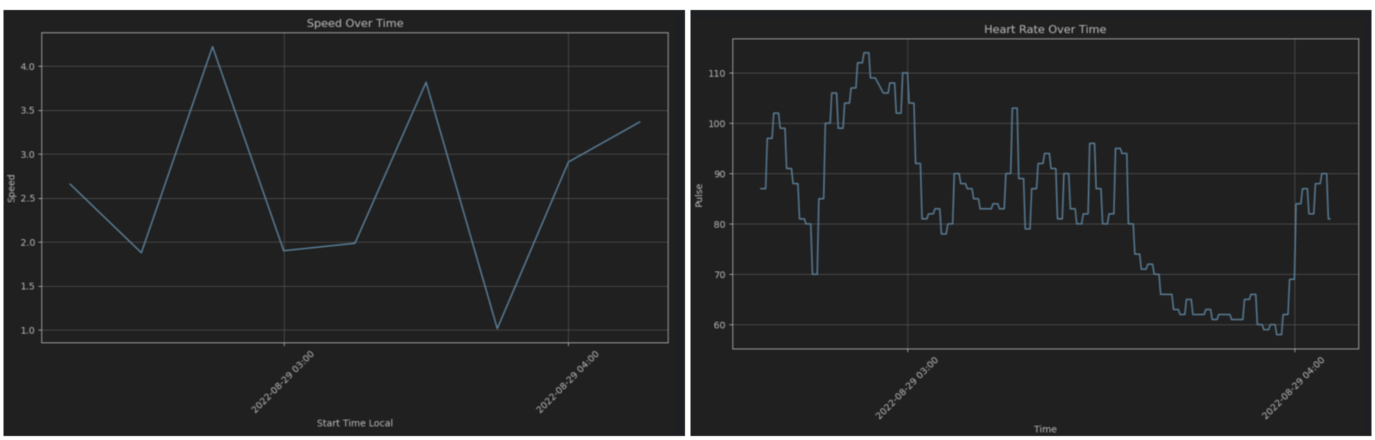
Correlation between speed and heart rate:

For a sanity check, we wanted to check if there is a correlation between speed and heart rate. We expect to see an increase in heart rate with an increase in speed, as physical exertion typically leads to a higher heart rate. As shown in the graphs, the heart rate does increase in response to higher speed, though with a slight delay. This delay is consistent with the normal physiological response, where heart rate lags behind the onset of physical exertion. Therefore, our graphs show the expected correlation between speed and heart rate, confirming the relationship we anticipated. The following are example from a couple soldiers over a certain time frame.

Example 1:



Example 2:



Sleep

Our objective is to organize the sleep datasets into weekly grids for each soldier with 1 minute time intervals. We broke up this objective into two main steps. First, we generated the sleep grids, populating them with available data and marking unknown timeslots with '?'. Next, we imputed the unknown timeslots using our heart rate grids.

Generating the sleep grids:

A notebook showing the sleep grids generation is in *create_sleeping_grid.ipynb*.

The sleep datasets are made up of three separate datasets, each one describing the timeslots of when a soldier was *awake* or in *light sleep* or in *deep sleep*. There are 42,473 entries in the awake dataset, 112,288 entries in the light sleep dataset, and 47,803 entries in the deep sleep dataset. The following is an example the awake data, and the other two datasets have the same format.

	userId	summaryId	startTimeInSeconds	endTimeInSeconds
0	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-631e4674-594c	1662933540	1662933780
1	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-631e4674-594c	1662933840	1662933900
2	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-631e4674-594c	1662933960	1662934080
3	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-631e4674-594c	1662934140	1662936660
4	00a7a796-c572-44d7-a950-7f6bca4a4394	x48cc04e-631e4674-594c	1662942660	1662942720

- startTime and endTime are in Unix timestamp format.

We generated our sleep grids to have a similar format to the previous grids. Rather than only filling in the grids with available data, we distributed it across the relevant dates and filled any missing timeslots with '?' to indicate unavailable data.

```
sleeping_grids = {
    userID: {
        week_number: pd.DataFrame(Date, Hour, Minute, SleepState, WeekNumber)
    }
}
```

Explanation:

- *sleeping_grids* is a dictionary keyed by userId.
- *sleeping_grids* [*soldierID*] is a dictionary keyed by week number.
- *sleeping_grids* [*soldierID*][*week_num*] is a dataframe containing the corresponding generated sleeping grid for the soldier in week_num.

Example showing '?' added where data was not available:

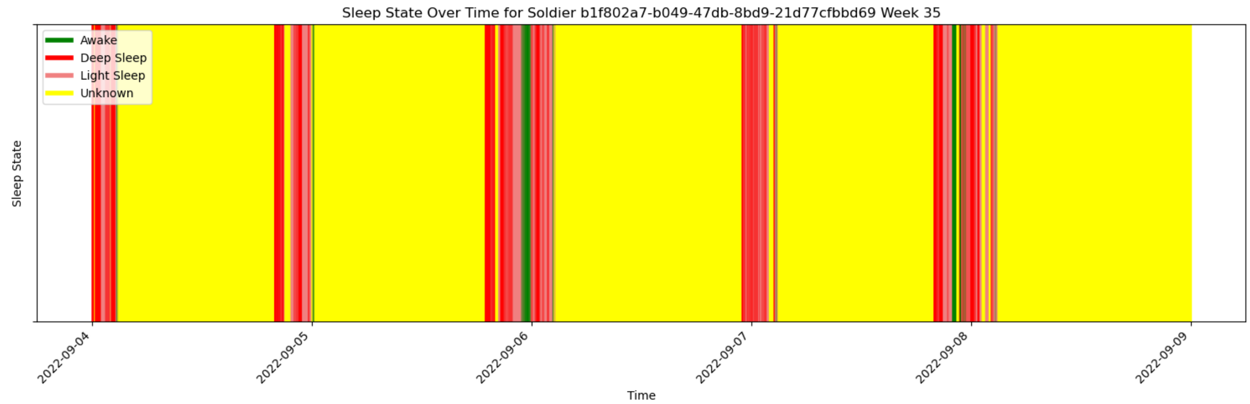
```
sleeping_grids['b1f802a7-b049-47db-8bd9-21d77cfbbd69'][44].head(15)
```

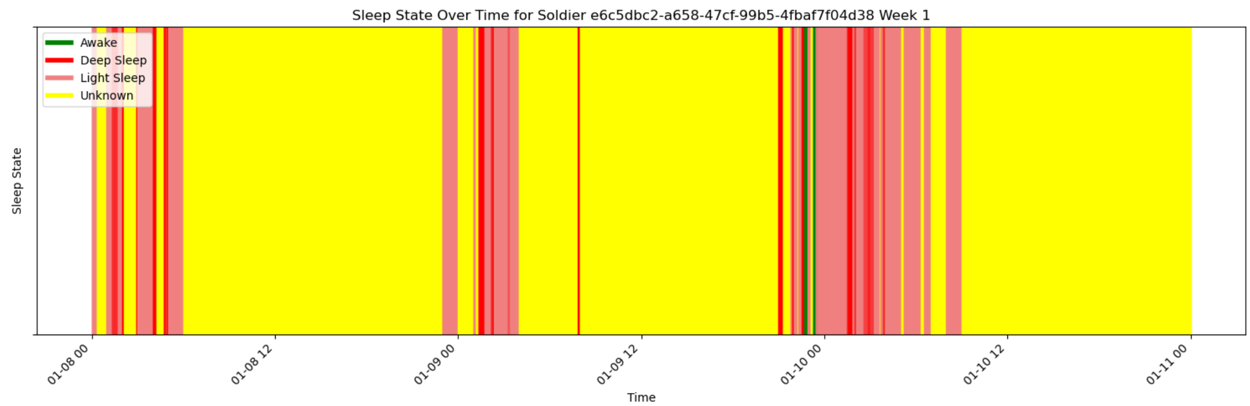
Executed at 2024.10.03 20:28:14 in 26ms

15 rows × 5 columns

	Date	Hour	Minute	SleepState	WeekNumber
56304	2022-11-06	0	0	Light Sleep	44
56305	2022-11-06	0	0	Deep Sleep	44
56306	2022-11-06	0	1	Light Sleep	44
56307	2022-11-06	0	2	Light Sleep	44
56308	2022-11-06	0	3	Light Sleep	44
56309	2022-11-06	0	4	Light Sleep	44
56310	2022-11-06	0	5	Light Sleep	44
56311	2022-11-06	0	6	Light Sleep	44
56312	2022-11-06	0	7	Light Sleep	44
56313	2022-11-06	0	8	Light Sleep	44
56314	2022-11-06	0	9	Light Sleep	44
56315	2022-11-06	0	10	?	44
56316	2022-11-06	0	11	?	44
56317	2022-11-06	0	12	Light Sleep	44
56318	2022-11-06	0	13	Light Sleep	44

The following are example visualizations of soldier's sleep states over a week. The yellow parts represent the '?' and would need to be imputed.





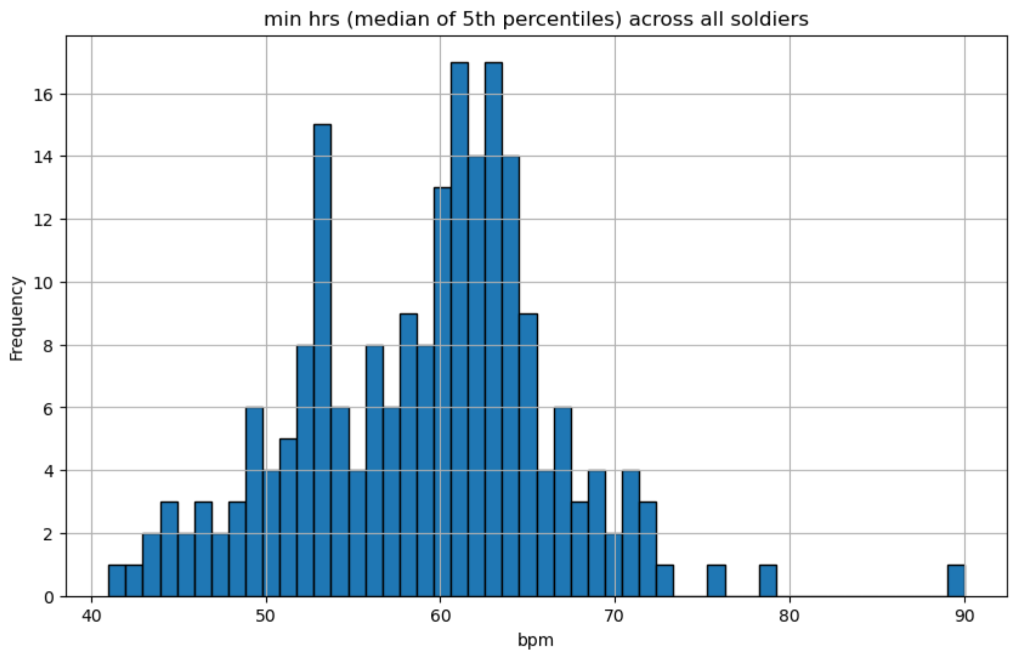
Sleep grids imputation:

A notebook showing the sleep grids imputation is in *completing_sleep.ipynb*.

Analyzing the sleep data reveals that soldiers primarily measured their sleep state during nighttime. However, since soldiers may also sleep during the day, we need a way to identify daytime sleep. To address this, we determined each soldier's minimum heart rate, which can serve as a personalized threshold to distinguish between sleep and light physical activity.

- *Min_hr*(x)= median{ [5th percentile hr for every recorded day by x] }

The following graph shows the distribution of the min heartrates across all soldiers.

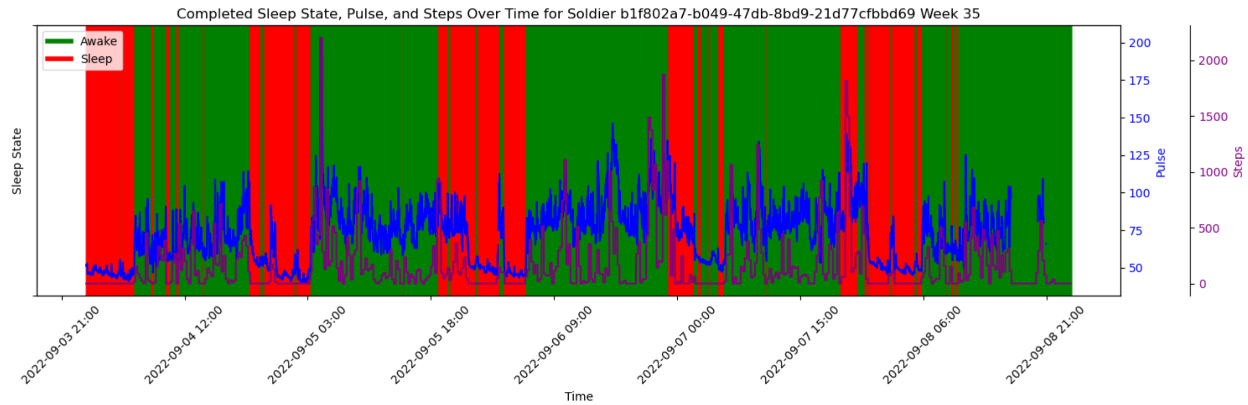


The following rule was applied to impute missing sleep data:

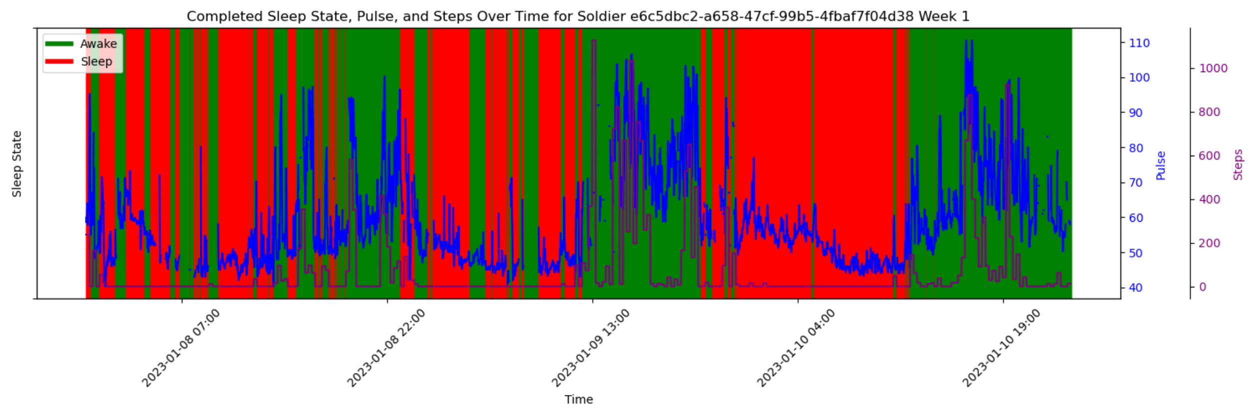
1. During night hours (19:00 to 06:00):
   - If HR ≤ min_hr, then SleepState ← 'Sleeping'
2. During day hours (06:00 to 19:00):
   - if steps == 0 and HR < 1.15*min_hr, then SleepState ← 'Sleeping'
3. If 5 rows in a row are 'Deep Sleep'/'Light Sleep'/'Sleeping', then the following row is 'Sleeping'
4. Replace remaining '?' as 'Awake'
5. Replace 'Light Sleep' and 'Deep Sleep' as 'Sleeping'

Note: in code, 'Sleeping' is 1 and 'Awake' is 0.

The following visualizations show the imputation applied to the two previous examples



- min hr of soldier is 52



- min hr of soldier is 46

To apply the imputation rule described above, the heart rate, steps, and sleep grids must all have consistent time intervals. Since the steps grids are recorded at 15 minute intervals, we upsample them by propagating each measurement across the corresponding 1 minute intervals. For the heart rate grids, which are recorded at 15 second intervals, we downsample by calculating the mean heart rate for each 1 minute interval.

The following is an example of downsampling heart rate:

15 second intervals

	userId	datetime	pulse
0	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:50:15	87
1	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:50:30	87
2	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:50:45	87
3	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:51:00	87
4	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:51:15	87
5	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:51:30	87
6	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:51:45	87
7	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:52:00	87
8	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:52:15	94
9	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:52:30	94
10	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:52:45	94
11	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:53:00	94
12	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:53:15	82
13	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:53:30	82
14	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-11 11:53:45	82

downsampled 1 minute intervals

	datetime	pulse
0	2022-09-11 11:50:00	87.00
1	2022-09-11 11:51:00	87.00
2	2022-09-11 11:52:00	92.25
3	2022-09-11 11:53:00	85.00

The following is an example of upsampling steps.

15 minute intervals

	userId	startTimeLocal	steps
0	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-04 09:15:00	49
1	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-04 09:30:00	103
2	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-04 09:45:00	29
3	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-04 10:00:00	172
4	00a7a796-c572-44d7-a950-7f6bca4a4394	2022-09-04 10:15:00	169

Upsampled 1 minute intervals

	startTimeLocal	steps
0	2022-09-04 09:15:00	49
1	2022-09-04 09:16:00	49
2	2022-09-04 09:17:00	49
3	2022-09-04 09:18:00	49
4	2022-09-04 09:19:00	49
5	2022-09-04 09:20:00	49
6	2022-09-04 09:21:00	49
7	2022-09-04 09:22:00	49
8	2022-09-04 09:23:00	49
9	2022-09-04 09:24:00	49

Imputation Statistics and Visualizations:

Note: My approach to sleep imputation differed slightly from Barak's method. For example, we calculated *min_hr* differently and Barak didn't use steps in his imputation method. In the end, we decided to use Barak's imputed sleep data.

We wanted to understand better how we imputed the sleep data. To do this, we calculated several statistics for each soldier and averaged them across all soldiers. Below are the specific statistics we computed, along with examples for each. Detailed calculations can be found in the notebook *Soldier_Statistics.ipynb*.

*day** is from 6am to 7pm. *123abc* is not an actual soldier, but just an example.

soldier_stats_preprocessing:

Soldier	Amount of data in minutes	Amount of filled sleep data	Amount of filled awake data	Amount of missing data	Percent of data filled	Percent of data missing	Amount of filled sleep during the day	Percent of day* filled in as sleep
123abc	2880	500	120	2260	21.53	78.47	100	12.82
…	…	…	…	…	…	…	…	…
average								

soldier_stats_postprocessing:

Soldier	Amount of data in minutes	Amount of filled sleep data	Amount of filled awake data	Amount of missing data	Percent of data filled	Percent of data missing	Amount of filled sleep during the day	Percent of day* filled in as sleep
123abc	2880	1400	1000	480	83.33	16.67	250	32.05
…	…	…	…	…	…	…	…	…
average								

soldier_imputation_stats:

Soldier	Amount of data in minutes that needs to be imputed (missing data)	Amount of missing data imputed as sleeping	Amount of missing data imputed as awake	Amount of data that was not able to be imputed	Percentage of missing data that was imputed as sleeping	Percentage of missing data that was imputed as awake	Percentage of missing data that was not able to be imputed	Percentage of overall data that is imputed
123abc	2260	900	880	480	39.82	38.94	21.24	61.81*
…	…	…	…	…	…	…	…	…
average								

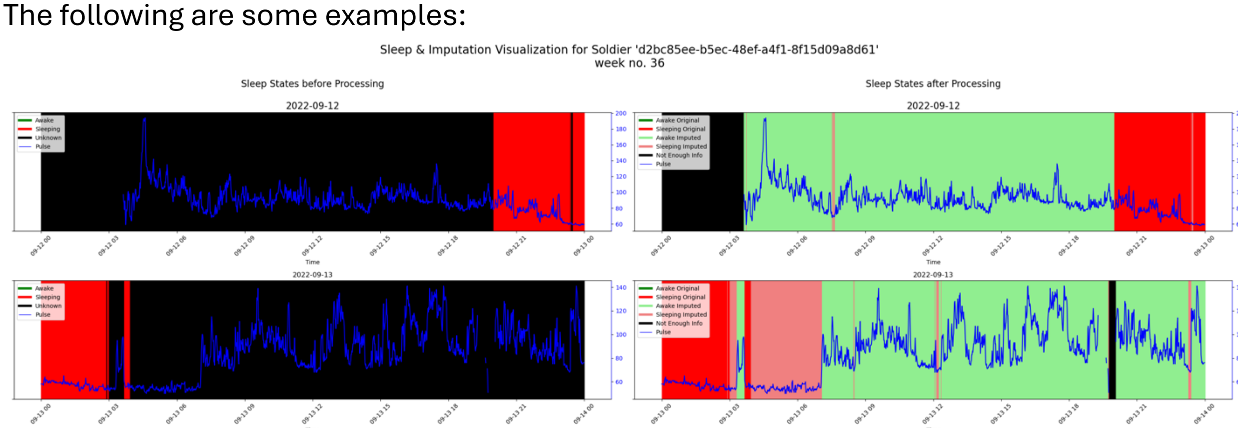
* $\frac{900+880}{2880} = 0.6181$

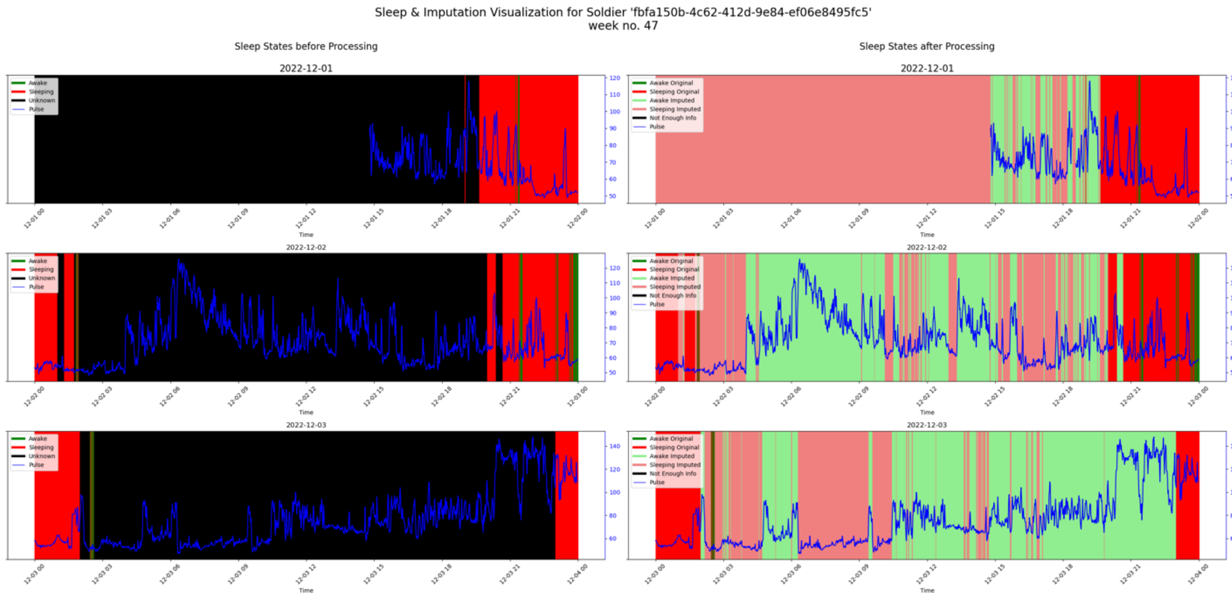
The following is a comparison of the averages pre and post sleep imputation.

	pre processing averages	post processing averages
Amount of data in minutes	78598.03	78598.03
Amount of filled sleep data	15034.03	41789.06
Amount of filled awake data	610.66	35771.93
Amount of missing data	62953.34	1037.05
Percent of data filled	18.44	95.11
Percent of data missing	81.56	4.89
Amount of filled sleep during the day	977.43	17422.36
Percent of day filled in as sleep	2.55	37.71

We also created a visualization for each soldier to demonstrate specifically what was imputed. The code can be found in *All_Soldiers_Sleep_Vis.py*.

The following are some examples:





Current State and Future Work

At this stage of the project, the preprocessing steps have been completed. We have organized the heart rate, steps, and sleep data into grids that are ready to be used in a deep learning model. Currently, I am preparing to start training an Autoencoder to generate a latent space of the heart rate data, with the goal of separating the latent space into two distinct groups. This will provide deeper insights into heart rate patterns and their correlation with potential injuries.

Conclusion

This project provided valuable experience working with real world, messy data - a stark contrast to the clean and structured datasets I've encountered in previous machine learning tasks like MNIST, CIFAR10, and TACO. In any project that uses real world data, the preprocessing phase is especially critical, because handling incomplete and inconsistent data from multiple sources is essential for the success of the model. Throughout this project, I gained significant insights into the preprocessing phase of building a ML model. Moving forward, these skills will be instrumental as I apply the skills I learned to other projects.