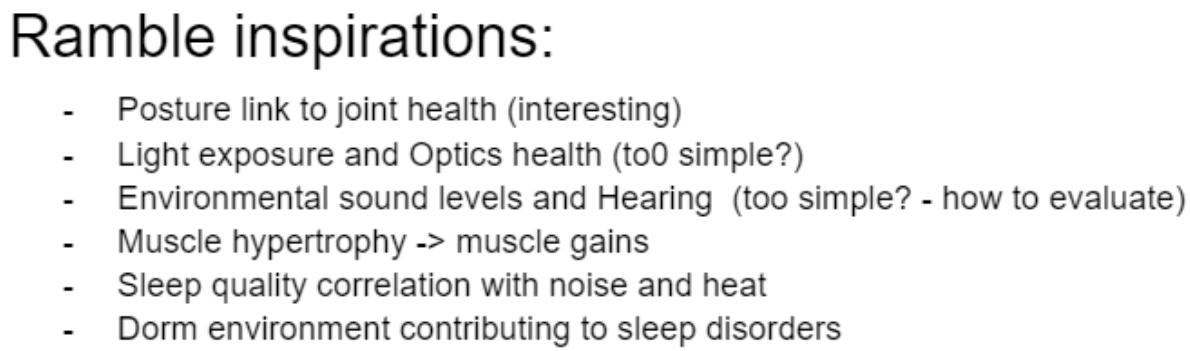
NIEjr04: Exploring Local Micro-Climates With Open-Source Sensors

This log will detail the timeline of my research journey with the hurdles and lessons along the phases of the project.

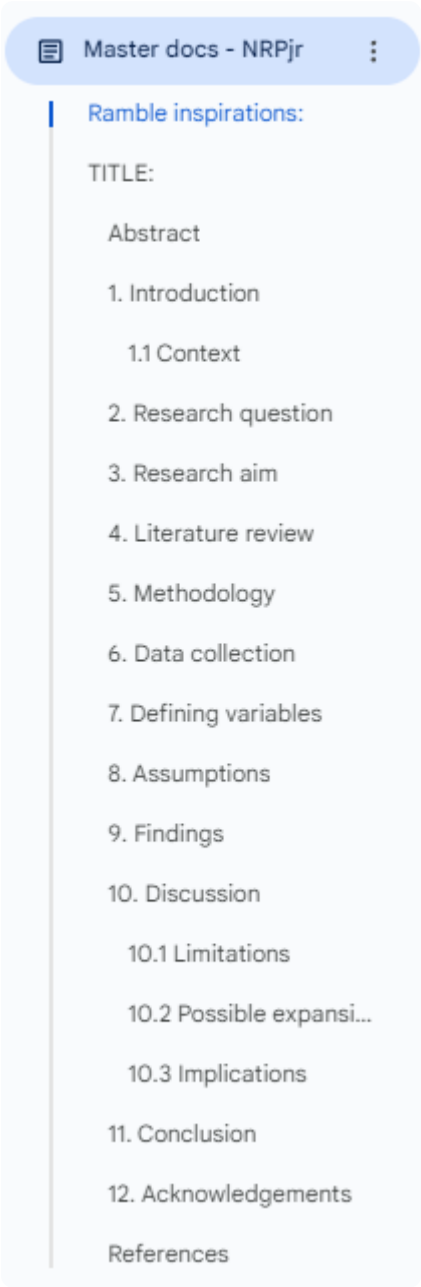
Planning phase

April - Outlining of project and choosing focus

- Beginning half: the commencement of NRPjr -> Phong and I get together to brainstorm rough ideas that we can expand on from the broad title code.



- 19th: Me and my partner meet up with our superviser Mr. Kenneth in person to discuss how we should procceed. We also brought up brainstormed ideas and got consulted them. He also shared about the procedures of the competition with deadlines and targets.
    - Actions points of the meeting:
        - Beginning writing out report stucture



        - Contacting the teacher in charge of equipment from Mr. Ken's team - Mr. Ahmed.
        - Phong will be researching past publications regarding similar topics. to aid in data analysing,
        - I will be researching implementation of the sensors, learning them and requesting their purchase from Mr. Ahmed

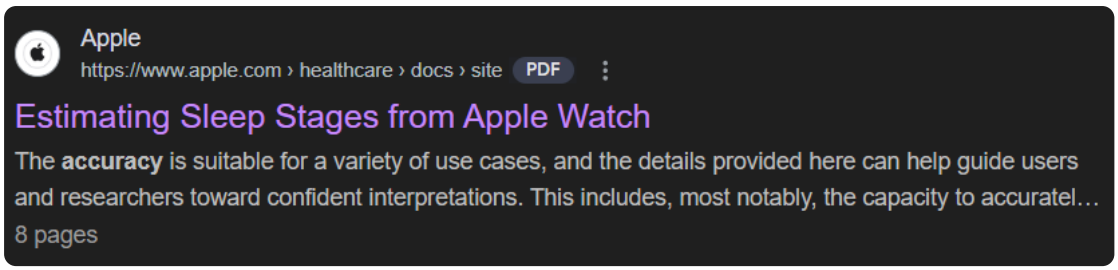
        - We narrowed the research aim to analysing sleep stages and quality
    - Our goals is learn as much as possible about metrics of sleep quality and possible ways we can measure them using open-source sensors.
    - We need to come up with a final thesis statement and plan our research

- End of month: I started learning arduino on my own with prior knowledge of c++, and some parts I borrowed from a friend.

Researching phase

May until mid-July - Main inspirations from lit-review

- In May, both Phong and I was only working up our knowledge of the specialty and of coding.
- Resources I investigated and learned from(for coding and for sleep medicine):
    - Sleep quality assessment using a standardised questionaire - Pittsburgh Sleep Quality Index (PSQI). Use of statistical models to correlate with sociodemographic, and anthropometric data, obstetric information, lifestyle habits, and sleep hygiene practices
        - Gomes, M.R.d., Rodrigues, J.C.M., Barbosa, L.M.A. et al. Factors associated with sleep quality in adolescent pregnant women. Sleep Breath 29, 54 (2025). https://doi-org.remotexs.ntu.edu.sg/10.1007/s11325-024-03205-y (https://doi-org.remotexs.ntu.edu.sg/10.1007/s11325-024-03205-y)
    - Suggested to us to read due to similar equipments of open-source sensors for environmental factors
        - Lee, J., Yu, B., Lim, K. (2023). Exploring The Implementation of AI in a Cost-effective Device for Predicting Sleep Quality. In: Lucas Paletta, Hasan Ayaz and Umer Asgher (eds) Cognitive Computing and Internet of Things. AHFE (2023) International Conference. AHFE Open Access, vol 73. AHFE International, USA. http://doi.org/10.54941/ahfe1003981 (http://doi.org/10.54941/ahfe1003981)
    - Apple's in-house sleep tracking accuracy evaluation:



- Public evaluation of wearables' sleep tracking abilities against **Polysomnography(PSG)**:
  - Lee T, Cho Y, Cha KS, Jung J, Cho J, Kim H, Kim D, Hong J, Lee D, Keum M, Kushida CA, Yoon IY, Kim JW. Accuracy of 11 Wearable, Nearable, and Airable Consumer Sleep Trackers: Prospective Multicenter Validation Study. JMIR Mhealth Uhealth. 2023 Nov 2;11:e50983. doi: 10.2196/50983. PMID: 37917155; PMCID: PMC10654909.
  - Miller DJ, Sargent C, Roach GD. A Validation of Six Wearable Devices for Estimating Sleep, Heart Rate and Heart Rate Variability in Healthy Adults. Sensors (Basel). 2022 Aug 22;22(16):6317. doi: 10.3390/s22166317. PMID: 36016077; PMCID: PMC9412437.
- PSG sleep stages used to match up with sleep posture:
  - Hatice Kutbay Özçelik, Mehmet Bayram, Emine Doğanay, Levent Kart. Effects of body position on sleep architecture and quality in subsyndromal adults without apparent obstructive sleep apnea Yakar https://doiorg.remotexs.ntu.edu.sg/10.1111/sbr.12116 (https://doiorg.remotexs.ntu.edu.sg/10.1111/sbr.12116)

Late July to August - Initial setup of data collection

- We contated Mr Ahmed to discuss the project.
- 5th of August we met up face to face to go over the equipments we'd need for the project. We also fill in the project details and progress so far for him.
- A week later we picked up the equipments Mr Ahmed sent over to our dorm and set up as the following instruction: https://docs.google.com/document/d/1iJ5y1pXXhdGfy8_7mez45kYNSWXKWL6B_d-jw0NOnyY/edit?usp=sharing (https://docs.google.com/document/d/1iJ5y1pXXhdGfy8_7mez45kYNSWXKWL6B_d-jw0NOnyY/edit?usp=sharing)

September to December - Hiccups in setting up

- We intended to use only the seeduino wio terminal unit to collect data and log data out into a micro SD cards. But later testing found that the equipments do not quite function as intended and we formulated a new arrangement and new equipments had to be bought - final arrangements shown in report.
- Discussion logs with Mr. Ahmed regarding setup hiccups: https://docs.google.com/document/d/1kizx409GpZcOY5VnxNFBB6-i69fI5ZomP6G5lUilYC4/edit?usp=sharing (https://docs.google.com/document/d/1kizx409GpZcOY5VnxNFBB6-i69fI5ZomP6G5lUilYC4/edit?usp=sharing)
- At the end we used Coolterm to log the data out through the serial output into a text dump.
- We change from using the i2c extension to using only the 2 default Wio ports and buying an additional Arduino uno unit with a Base shield v2

Execution phase

December - results of research

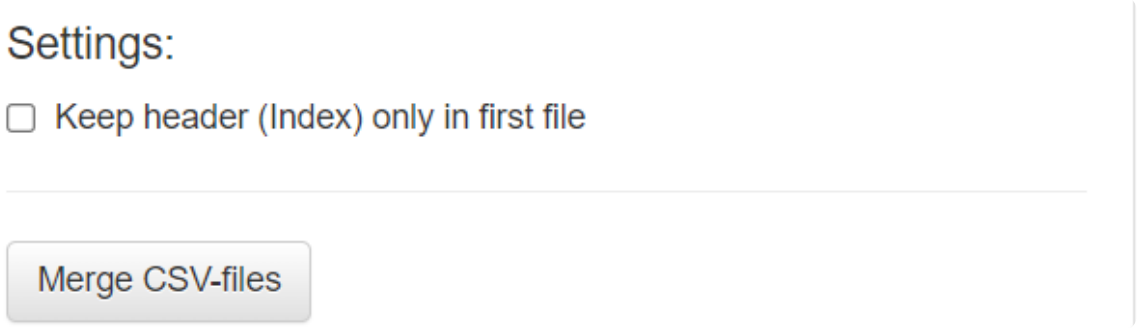
- Our data collection kicked off in 12-12-2024 and our quota is 14 sleep sessions, totaling 2 weeks with the last night being Christmas night - 25 Dec.
- We use convert the environmental data into .csv formatting and the Apple health sleep data is extracted using a free app: Simple Health Export CSV on appstore.
- We plan to clean up, process and analyse the data using basic python programming and the IBM SPSS 27 software.

Data processing hurdles:

"basic python programming" turned out to be not so basic.

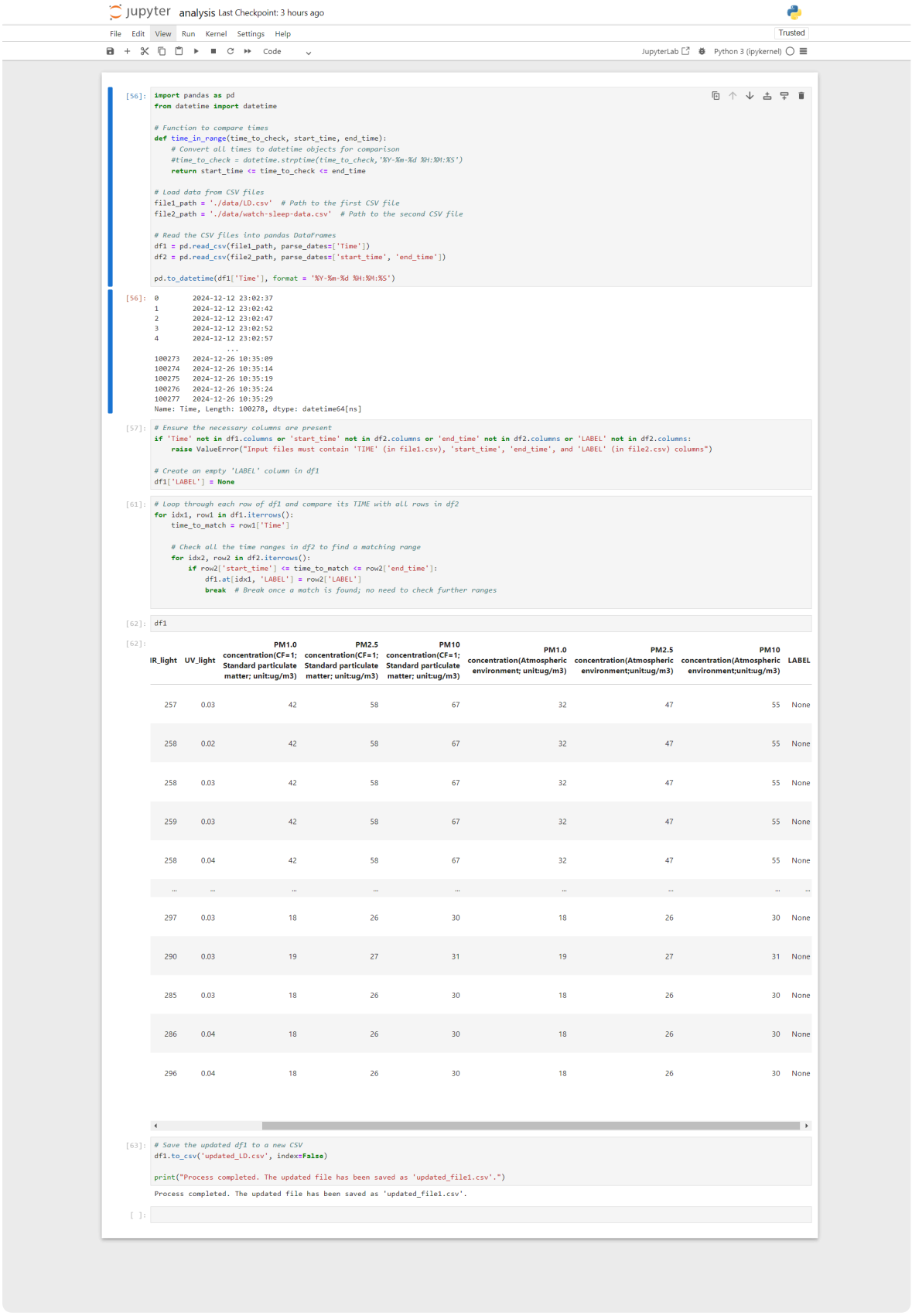
- For apple data:
    - I tried to use a third party app: simplehealhexportCSV BUTTT it turns out the time zone was fickly as i was sharing an apple id with someone abroad.
    - Then I struggled for awhile before realising I can export via apple health itself, I just needed to sieve through the .xml file to find the sleep data as the file is a dump of all kinds of data.

    - Howeverrrr, remember the format is roughly this: start, end, sleep_stage. And the sensors are such: timestamp, value, value,... HOW DO WE COMBINE THESE FORMATS???
- For the environmental sensors:
    - The data was collected as such: each day of sleep is a text file in roughly CSV format, A simple script c++ to add commas and renaming to .csv did the job
    - I then just need to merge the files vertically which, there is a website for that; so... it's all good right??? (there was an suspicious little checkbox that i forgot to tick...(foreshadow))



    - I tried to also merge the 2 environmental sensors together because they are essentially the same js seperate for operational reasons but quickly stopped because they obviously dont quite match up.
- Combining the 2 HOWWW???!!!
    - My idea from reading other researches is to categorise the environmental sensors readings with the time ranges of the sleep stages.
    - Essentially each timestamp of the sensors is cross-referenced with all the ranges of the apple watch. each timestamp is then categorised: awake, inbed, rem, core, deep according to the time range they are sandwiched in.

- SEEMSSSS doable? I know I could do it on pandas-python.



- As you can see, it ... diddd wOrKs? But trust me i forgot to screenshot but the elasped time was around 30 minute and it lost some data(in hindsight)
- SO I WENT AHEAD AND CODED A C++ SCRIPT, SCRAPING THE FILE, PAINSTAKINGLY STRING PROCESSING AND IT ACTUALLY RAN BOTH

SENSOR FILES BEFORE THE PYTHON FINISHES.

```cpp
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <ctime>
#include <iomanip>
#include "utils/dbg.h"

// Helper function to convert a date-time string to std::tm
std::tm stringToTime(const std::string& timeStr) {
    std::tm tm = {};
    std::istringstream ss(timeStr);
    ss >> std::get_time(&tm, "%Y-%m-%d %H:%M:%S");
    return tm;
}

// Helper function to compare two times
bool isTimeInRange(const std::tm& time, const std::tm& start, const std::tm& end) {
    std::time_t time_t_time = std::mktime(const_cast<std::tm*>(&time));
    std::time_t time_t_start = std::mktime(const_cast<std::tm*>(&start));
    std::time_t time_t_end = std::mktime(const_cast<std::tm*>(&end));

    return (time_t_time >= time_t_start && time_t_time <= time_t_end);
}

struct DataRow {
    std::string Time;
    std::vector<std::string> other_entries;  // Holds other 9 entries
    std::string Label;
};

struct TimeRange {
    std::tm start_time;
    std::tm end_time;
    std::string label;
};

int main() {
    // File paths for input CSVs
    std::string file1 = "THS.csv";  // First file (with 'Time' and 9 other columns)
    std::string file2 = "watch-sleep-data.csv";  // Second file (with 'start_time', 'end_time', and 'LABEL')

    // Read the first CSV file (df1) into a vector of DataRow
    std::vector<DataRow> df1;
    std::ifstream infile1(file1);
    std::string line;
    std::getline(infile1, line);  // Skip the header line
    while (std::getline(infile1, line)) {
        std::istringstream ss(line);
        std::string timeStr;
        std::getline(ss, timeStr, ',');  // Get 'Time' column

        DataRow row;
        row.Time = timeStr;

        // Get the next 9 entries (other columns)
        for (int i = 0; i < 3; ++i) {
            std::string entry;
            std::getline(ss, entry, ',');
            row.other_entries.push_back(entry);
        }

```

```
        df1.push_back(row);
    }
    infile1.close();

    // Read the second CSV file (df2) into a vector of TimeRange
    std::vector<TimeRange> df2;
    std::ifstream infile2(file2);
    std::getline(infile2, line);  // Skip the header line

    while (std::getline(infile2, line)) {
        std::istringstream ss(line);
        std::string start_time_str, end_time_str, label_str;

        // Skip the first 4 unused columns
        for (int i = 0; i < 4; ++i) {
            std::string unused;
            std::getline(ss, unused, ',');
        }

        // Read start_time, end_time, and label from the CSV
        std::getline(ss, start_time_str, ',');
        std::getline(ss, end_time_str, ',');
        std::getline(ss, label_str, ',');
        // Skip the final unused column
        std::string unused_end_col;
        std::getline(ss, unused_end_col, ',');

        std::cout << start_time_str << '\t' << end_time_str << std::endl;
        TimeRange range;
        range.start_time = stringToTime(start_time_str);
        range.end_time = stringToTime(end_time_str);
        range.label = label_str;
        df2.push_back(range);
    }
    infile2.close();

    // ! dsfsd

    // Now compare each Time in df1 with all time ranges in df2
    for (auto& row : df1) {
        std::tm timeToMatch = stringToTime(row.Time);
        bool matched = false;

        // Compare with each time range in df2
        for (const auto& range : df2) {
            if (isTimeInRange(timeToMatch, range.start_time, range.end_time)) {
                row.Label = range.label;
                matched = true;
                break;  // Once we find a match, no need to check further ranges
            }
        }

        // If no match is found, we can optionally set a default label
        if (!matched) {
            row.Label = "No Match";
        }
    }

    // Write the updated data to a new CSV file
    std::ofstream outfile("THS_with_labels.csv");
    outfile << "Time";
    for (int i = 0; i < 3; ++i) {
        outfile << ",Column" << (i + 1);
    }
    outfile << ",Label\n";

    for (const auto& row : df1) {
```

```
        outfile << row.Time;
        for (const auto& entry : row.other_entries) {
            outfile << "," << entry;
        }
        outfile << "," << row.Label << "\n";
    }

    outfile.close();

    std::cout << "CSV processing completed successfully. Output saved to 'df1_with_labels.csv'.\n";
    return 0;
}
```

In the end, on 26/12/2024 I got the files to an analysable format of timestamps with sensor readings + sleep stage labels. YAYYYY!!!
"If you're going through hell, keep going."
— Winston Churchill

- sleep stages order according to apple's sleep tracking publication (https://www.apple.com/healthcare/docs/site/Estimating_Sleep_Stages_from_Apple_Watch_Sept_2023.pdf (https://www.apple.com/healthcare/docs/site/Estimating_Sleep_Stages_from_Apple_Watch_Sept_2023.pdf)):

Apple Sleep State	Description
REM	Rapid eye movement sleep
Deep	Stage N3 or slow-wave sleep
Core	Stages N1 and N2 sleep

  - inbed < awake < core < deep < rem -> ordinal data?

Statistical analysis commenced

- 27/12/2024: The following document is my planning for the analysis, detailing the hypothesis(es??) and the techniques used. https://docs.google.com/document/d/1U3JP_9D7MFvbUrCdlgKKLPOmuKH_V7fgGfG0rS1rZy0/edit?usp=sharing (https://docs.google.com/document/d/1U3JP_9D7MFvbUrCdlgKKLPOmuKH_V7fgGfG0rS1rZy0/edit?usp=sharing)
  - Today the first two hypothesis are tested: the data do NOT really correlate with the label but the environmental factors are distinct when grouped by sleep stages, proving useful for third hypothesis: prediction

- 29/12/2024: Turns the data does not meet the assumption of proportional odds that is needed for the Ordinal regression to work. The parallel lines test had a p-value < alpha(0.05).
    - SO NOW WE TRY Multinomial Logistic Regression!.. ermmm doesnt seem plausible

Finalising phase

January - Submission

- Writing the report and touching up the rough edges.