TGO MODULE 12 2023

THE CORRELATION BETWEEN OUTCOME MEASURES OF SMARTWATCH DATA AND MDS-UPDRS PART-III SCORES IN PARKINSON'S DISEASE PATIENTS

A research which analyses the outcome measures of smartwatch data and MDS-UPDRS Part-III scores such as tremor and dyskinesia. Including a literature search of Parkinson's disease and an update of useful upcoming techniques who can improve the current treatment.

June 26, 2023

Technical Medicine Bachelor.
TGO group 8: Job Meuwese (s2464624),
Loet Schoenmakers (s2506327) and Stef Berendsen (s2507781).
University of Twente.
Amsterdam UMC, Department of Neurology.

Under the guidance of: M.Beudel, M.G.J. de Neeling,
Dr.ir. T. Heida, MSc. B.J.C.C. Sweep and T.F. Ruuls.

Abstract

Parkinson's disease (PD) is the second most common neurodegenerative disease, and the number of Parkinson's patients is expected to double in the coming years [1]. The increase in the number of patients could cause problems with the current treatment method. Parkinson's disease cannot be cured, the symptoms can only be reduced. This can be done through medication or deep brain stimulation (DBS). The Amsterdam UMC is working on a study that should ensure that deep brain stimulation becomes adaptive (aDBS). So that the patient and/or doctor can immediately change the strength and amount of DBS. The study uses smartwatches to detect tremor and dyskinesia and an MDS-UPDRS scoresheet to consider current status.

This research provides a first set-up for processing and analysing the smartwatch and MDS-UPDRS data. In addition, research has been conducted into whether there is a correlation between the smartwatch data and the MDS-UPDRS score list. This eventually led to the following research question: What is the correlation between outcome measures of the smartwatch data and MDS-UPDRS scores of patients with Parkinson's disease?

With the help of a literature search, the researchers have formed a basis about Parkinson's disease. But also about current techniques such as machine learning and artificial intelligence to see whether they can be combined with current and future treatment methods. After this, scripts were written using Python to process, analyse and correlate the available data from smartwatches and MDS-UPDRS.

Using 3 iterations of a Random Forest (RF) algorithm, the RF had average accuracies ranging from 48,9% to 52,7%. Accuracies for values within one of the correct value ranged from 80,9% to 84,8%. Next to this, data on probabilities of tremor and dyskinesia, MDS-UPDRS Part-III scores and time availability were plotted. With this retrieved data and algorithms, we concluded there to not be a correlation between smartwatch data and MDS-UPDRS Part-III scores.

Contents

1 List of abbreviations

Abbreviation	Definition
aDBS	Adaptive deep brain stimulation
AI	Artificial intelligence
Amsterdam UMC	Amsterdam University medical centre
DBS	Deep brain stimulation
DT	Decision tree
D1 receptors	Dopamine receptor subtype-1
D2 receptors	Dopamine receptor subtype-2
EEG	Electroencephalography
EMG	Electromyography
GABA	Gamma-aminobutyric acid
GPe	Globus pallidus externus
GPi	Globus pallidus internus
IMU	Inertial measurement unit
MDS-UPDRS	Movement Disorder Society- Unified Parkinson's Disease Rating Scale
ML	Machine learning
MM4PD	Motor fluctuations Monitor for Parkinson's Disease
MSN	Medium spiny neurons
NN's	Neural networks
PD	Parkinson's disease
QoL	Quality of life
RF	Random forest
SNc	Substantia nigra pars compacta
SNr	Substantia nigra pars reticulata
SNR	Signal to noise ratio
STN	Subthalamic nucleus
SVM's	Support vector machines
vGLUT2	Glutamate transporter isoform 2

2 Introduction

Parkinson's disease is the second most common neurodegenerative disease, and the number of Parkinson's patients is expected to double in the coming years [1]. Parkinson's is a progressive disease in which motor and non-motor complaints are experienced. The status of the disease is determined by a scoring list called Movement Disorder Society-Unified Parkinson's Disease Rating Scale (MDS-UPDRS). Current treatments exist of different types of medication and in a later stadium deep brain stimulation (DBS). In the hospital, the strength of the stimulation can be adjusted based on the symptoms the patient experienced in the prior weeks. But would it not be easier to adjust the stimulation based on the symptoms and patients experiences at that moment? And how can you determine the right moment to apply this adjustment?

The origin of the place where Parkinson's arises is well investigated including the complaints and treatment method [2, 3, 4]. However, the question of how it arises is still unknown. This paper also discusses the current and future expectations of DBS [5]. It appears that care for Parkinson's disease focuses on the use of DBS in combination with artificial intelligence and machine learning. This should ensure that care becomes more patient specific.

A study is being held at Amsterdam University medical centre (Amsterdam UMC) that specifically investigates adaptive deep brain stimulation (aDBS). The study has seven measurement points in the study where an MDS-UPDRS score sheet is taken. Two weeks after the surgery, the first measurement point is registered and then the assessment is taken every two months for up to half a year. During this time, the patient wears a smartwatch that uses the gyroscope and accelerometer to measure the percentage of time that tremor and dyskinesia are present (interval is one minute). aDBS must become an improvement of deep brain stimulation that ensures that an algorithm will sense when a patient needs a certain amount of stimulation. This algorithm will base this on the patient's specific profile, which consists of patient information and smartwatch movement data, and neural activity. The study itself consists of several parts such as classification of MDS-UPDRS, classification of smartwatch data and the correlation between the MDS-UPDRS and smartwatch data. The importance of this is that the data can be directly compared, so that the doctor and patient can see whether an adjustment of the adaptive deep brain stimulation system is necessary. These parts of the study are outsourced to us. Our research question is therefore:

What is the correlation between outcome measures of the smartwatch data and MDS-UPDRS scores of patients with Parkinson's disease?

To answer this research question, a literature study was first performed. To then compare and visualize the smartwatch-and-MDS-UPDRS data separately. Afterwards, these sets of data are compared with each other to find a correlation between them. The aim of this research is therefore to supply the basis for analysing smartwatch data and MDS-UPDRS scores. And provide insight into how they correlate.

The next chapter of this scientific report provides an overview of the current knowledge about the disease and technology. Subsequently, chapter 4 discusses the method of this study. In chapter 5 the results will be reflected upon and in chapter 6 we will discuss which factors had an influence and how the research could have been even better. Finally, in chapter 7 the conclusion will be discussed.

3 Theoretical background

3.1 Parkinsons disease

3.1.1 Introduction

The musculoskeletal system is a collaboration of multiple organs and tissue types. A disruption of this mechanism can have a major impact on the quality of life (QoL) of the patient. Due to several types of pathologies, disruptions can occur in the functioning of the musculoskeletal system. A common neurological disorder is Parkinson's disease (PD). Due to a disturbed signal transduction, the patient has less control over the contraction and relaxation of the muscles. This section will elaborate on which parts of the brain are involved with PD, how normal transduction proceeds and how PD influences these pathways.

3.1.2 Cerebrum

To get a better understanding of the motor neuron pathway, it is important to know which parts of the brain are involved. To start, the brain consists of four lobes: the frontal, parietal, temporal and occipital lobe. To focus on the part of the brain that controls the movement, we have to zoom in to the frontal lobe. The pathway starts in the motor cortex, this area can be subdivided in the primary and non-primary motor cortex [6]. This non-primary part of the brain region is made up of the premotor cortex and the supplementary motor area. The premotor cortex plays a key role in planning movements, based on sensory stimuli. This brain area will help to perform an accurate movement, when a sensory stimulus is received. In conclusion this brain region thus helps to combine the sensory stimulus and perform an adequate reaction. For patients in whom the premotor cortex functions less, the coordination between senses is lessened. For example, it will be more difficult for these patients to catch a ball. While observing the ball and lifting the arm (separately) is on the same level as healthy subjects. [7]

The supplementary motor area helps planning movements, like the premotor cortex. The difference between the functionality of these two, is the sensory input. While the premotor cortex is focused on recieving information from outside (senses), the supplementary motor area is primarily based on movements that have been performed during similar events.[8] In the frontal lobe against the central sulcus lies the primary motor cortex. This part of the brain is responsible for conscious movements. Each body part has its own area on the sagittal axis of the primary motor cortex. The neurons for controlling the facial muscles are located laterally, while the control of the hands and feet takes place medially. [9]

3.1.3 Basal Ganglia

The basal ganglia consists of several parts, including the caudate nucleus, the putamen and the accumbens nucleus. Together, these structures form the input nuclei. As output nuclei there are the globus pallidus in and externus (GPi and GPe) and the substantia nigra pars reticulata (SNr)[2]. The feedback loop consists

of three pathways: the direct, indirect and hyperdirect pathway, the hyperdirect pahtway does not use dopamine receptors and therefore plays little to no part in PD. Because while the direct pathway promotes movement, the indirect pathway inhibits movement. In the next paragraph we will give more detailed information about their working mechanisms, see figure 1 [2, 3]. These pathways consist of two types of neurons: the medium spiny neurons (MSN) and interneurons. The MSN make up 90 percent of all neurons in the striatum and are GABAergic inhibitory neurons of efferents. While the interneurons comprise 10 percent of all neurons in the striatum, they are mainly smooth dendrites and mainly use acetylcholine as a neurotransmitter. [2, 3]

The direct pathway consists of the following parts: the striatum, GPi and SNr. In this pathway, the striatal MSN's project directly to the GPi and SNr. They do this using dopamine receptor subtype-1 (D1 receptors)[3]. The subthalamic nucleus (STN) stimulates the cortex with signals to move. However, the GPi brakes this movement and ensures that unwanted movements are not made [3]. When we want to make a movement, the cortex sends stimulating signals to the striatum. The cortex does this through glutamate transporter isoform 2 (vGLUT2). The striatum consists mainly of GABAergic neurons that inhibit the action of GPi. Inhibiting GPi ensures that STN is less inhibited and can therefore deliver its excitation signals for movement. [3, 4]

The indirect pathway consists of the striatum, GPe, STN, GPi and SNr. This pathway utilizes the dopamine receptor subtype-2 (D2 receptors) [3]. The GPe consists mainly of GABAergic neurons and inhibits the action of STN. When the indirect pathway is activated, the cortex, as in the direct pathway, ensures that the striatum is stimulated by vGLUT2. The striatum then subsequently inhibits GPe through GABAergic neurons [4]. The GPe, like GPi in the direct pathway, inhibits the stimulation of STN excitatory signals. Only now does the cortex also directly stimulate the STN. This causes the STN to stimulate the SNr and GPi. The GPi and SNr ensure that inhibitory signals are delivered to the thalamus. So that the thalamus does not send excitatory signals to the cortex. However, the substantia nigra pars compacta (SNc) can now produce dopamine. An increase in dopamine in the striatum ensures that the effect of the indirect pathway is inhibited and that movement does take place. [3, 4]

The hyperdirect pathway is the fastest feedback loop of the three [10]. The cortex projects directly onto the STN. Which then activates the SNr and GPi. The striatum is thus skipped. The hyperdirect pathway is assumed to have the same effect as the indirect pathway. It incites movement. Research by C. Bingham et al.[10] shows that both the function and existence of the hyperdirect pathway are not yet clear.

Lanciego et al.[2] shows that in Parkinson's disease, the feedback loop of the basal ganglia changes. The SNc produces too little dopamine. This causes the STN to show more activity. Overactivity of the STN, in turn, causes the GPi and SNr to be overstimulated, leading to inhibition of the thalamus. This inhibits the projection of cortical neurons associated with inciting movement. [2]

Alfa-synuclein is a protein that appears to play an important role in neurotransmitter release and presynaptic function [12]. The protein is mainly located at the presynaptic terminal. In several diseases such as PD there are accumulations of this protein in different places in the body, these accumulations are also called Lewy bodies [12, 13]. How much and where these Lewy bodies are depends on a balance between

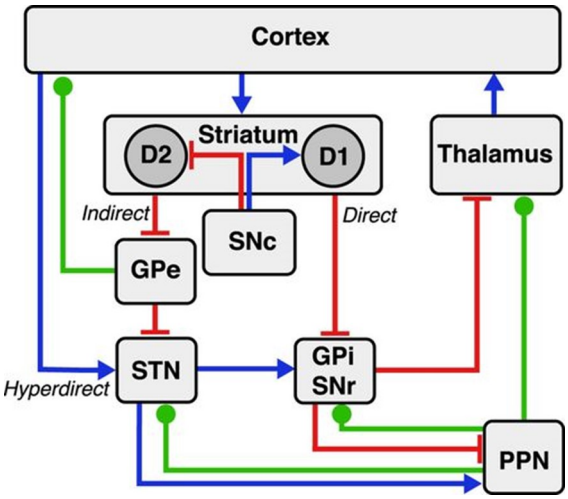

Figure 1: The direct and indirect pathway of the basal ganglia.
Note. Reprinted from "Mechanisms of deep brain stimulation" by T.M. Herrington, J. J. Cheng and E. N. Eskander, 2016, Journal of Neurophysiology. Copyright 2016 the American Physiological Society. [11]

clearance, synthesis and aggregation. Any changes in this balance can cause the amount of alfa-synuclein to increase. An increase in alfa-synuclein causes more (toxic) oligomers and fibrils to be produced [12]. The resulting fibrils change the permeability of nerve cells. This results in a greater influx of calcium ions, which in turn leads to cell death [12, 14]. The oligomers are harmful by damaging mitochondria or causing leakage into lysosomes. In Parkinson's disease, these Lewy bodies are found in the substantia nigra. The death of neurons in this place therefore ensures that too little dopamine is produced.

After the neurons have passed the Basal Ganglia, the nerve pulse passes through the cerebellum. In the cerebellum, there are sensory nerves that perceive the feedback of movement. This information eventually returns to the motor and sensory cortex. Damage to the cerebellum will impair a patient's fine motor skills. Due to the limited feedback, the signal strength can be adjusted less quickly in the motor cortex. [15, 16]

3.1.4 Neurotransmitters

To understand the pathology of Parkinson's disease, it is important to investigate the role of neurotransmitters in the disease process. Especially the imbalance in the presence of these signalling substances is important for research into possible therapies. The main neurotransmitters in Parkinson's disease are dopamine and gamma-aminobutyric acid (GABA) . Both neurotransmitters are decreased in quantity in the disease [17]. Dopamine decreases due to decreased activation of the D2 receptor. This receptor is responsible for the synthesis, release and uptake of dopamine [18]. At a later stage of the disease process, the D1 receptor also shows loss of function [17]. Medications that increase dopamine levels show good results because they partially take over the production of dopamine. The expression of GABA is reduced in this condition, the function of this neurotransmitter is to inhibit movements. If the concentration is too low, the patient will suffer from tremor and stiffness. [17]

3.1.5 Brain's bio-electrical activity

The activity of brain waves can be used to explain the physiology behind the symptoms. These brain waves can be divided in five groups. Firstly, the Delta rhythm has a frequency range of 0-4 Hz and can be measured during deep sleep when the brain has very low activity. The theta rhythm comes second in terms of frequency (4-7 Hz) and is also measured during sleep or when someone is drowsy. The third category is called the alpha rhythm (7-13 Hz) and is active when the brain is awake but in a state of relaxation. From the Beta rhythm (13-31 Hz) and further the brain is actively processing a task, this can be a physical task or a mental assignment. It's important to note that closing the eyes will suppress the beta frequency. The last domain is the Gamma range, this range starts at 32 Hz and can have a bigger amplitude when two sensory signals are processed, for example sound and sight. [19]

For patients suffering from PD, the amplitude of Beta frequencies are increased. This mechanism is responsible for the bradykinesia. Because higher amplitudes of Beta frequencies inhibit motion. In healthy individuals these range of frequencies are suppressed during movement to improve the mobility. To measure these brain waves physicians can perform an electroencephalogram (EEG) on the skull of the patient.

Patients with DBS have metal leads implanted, these have both a stimulating function as a sensory function. Thus with the implants, the brain activity close to the STN can be detected. The physician can use this information to evaluate the effectiveness of the DBS. Overstimulation or the combination of DBS with medication can cause dyskinesia. During dyskinesia the Beta frequencies are suppressed too much, which causes over-movement. To minimalize this side-effect, the DBS has to be turned down temporally after the patient has taken medication.

In the same study from E. Florin et al.[20], the researchers found an increase in the Gamma activity by patients with PD. This confirmed their hypothesis, as an increased amplitude of Gamma waves correlates with prokinetic activation and has to compensate the inhibitory features of the increased beta oscillation. [20]

3.1.6 Pathophysiology of Parkinson's disease

In Parkinson's disease, the production of dopamine is disrupted. The substantia nigra is responsible for the production of this neurotransmitter [21]. The substantia nigra and striatum together form the nigrostriatal system and have a crucial influence on controlled movement [22]. In patients with PD, the disease manifests itself in different ways. First, the complaints can be divided into motor and non-motor abnormalities. Motor problems can be tremors, rigidity and bradykinesia. The non-motor abnormalities are part of the prodromal phase of the disease, the main symptoms of this phase are REM sleep problems and anxiety or depression [23]. Also, a person's cognition may decline early on in PD. These omens can start up to 20 years before the onset of motor abnormalities. However, these complaints can also be caused by other diseases [23].

No blood tests or other imaging tests are available for diagnosis. Currently, doctors make the diagnosis based on medical history, symptoms, and neurological exams. These characteristics are compared with atypical symptoms of PD in order to conclude whether the diagnosis is correct or whether there is another (neuro)pathophysiology underlying the complaints [24].

One of the neurological examinations consists of taking scores. The most well-known are the MDS-UPDRS [25] and Parkinson's disease questionnaire (PDQ-39)[26]. A clinical expert will review this list with the patient. The patient can indicate how much he is bothered by certain complaints. These complaints are both non-motor and motor complaints. With these scores, the doctor can determine the severity of the PD and the current state of the complaints during the doctor's visit.

The prodromal complaints are insufficiently specific for a diagnosis of PD. Prodromal complaints are non-specific complaints but may indicate a pre-stage of PD [23]. The presence of prodromal complaints can be used in combination with knowledge about exposure to risk factors for the patient. Examples of these factors are genetic abnormalities or environmental factors. Certain environmental polluting factors influence the development of neural damage, as is the case in PD. Mainly toxic substances such as pesticides and other chemicals are risk factors, researchers also see a correlation between economic growth (more industry) and the prevalence of PD. [24]

3.2 Therapy

3.2.1 Medical therapy

As discussed earlier, there is a dopamine deficiency in PD and patients with PD can present a wide range of complaints. Currently, PD is irreversible [27]. Hence why the therapy is aimed at reducing the complaints of patients. Reducing these complaints can be done in different ways. The presence of complaints without medication is also called the “off period”. While if there are no complaints during medication it is called the "on period" [28]. The most commonly used drug for PD is levodopa: a dopamine precursor [28, 29]. Levodopa crosses the blood-brain barrier and here it is decarboxylated causing levodopa to become dopamine [29]. This drug is often used in combination with carbidopa. Carbidopa cannot cross the blood-brain barrier, so its action is mainly peripheral. Because the effect is peripheral, more levodopa can be converted into dopamine [28, 29]. Carbidopa thus inhibits the breakdown of levodopa. In addition, Carbidopa helps to suppress nausea, low blood pressure and restlessness [30]. If these two medicines are combined, the complaints can be greatly reduced. Nevertheless, levodopa also has side effects. There are dizziness and gastrointestinal problems.[28]

In addition, dopamine agonists are sometimes also given. These agonists perform the same role as dopamine once they bind to a dopamine receptor [28]. An advantage of these agonists is that they need to be administered less frequently than levodopa. Often agonists and levodopa are combined, this ensures that the effect lasts longer and that the strength of the effect is greater [28]. Another class of medication is

inhibitors. This class of medication inhibits/decreases the metabolism of levodopa. The agent is therefore broken down less quickly and then has a longer effect.[28]

An alternative treatment option is physiotherapy, more specifically physical exercise (Spinning) [24]. To investigate effectiveness, van der Kolk et al.[31] randomly divided patients into two groups; a group that did high-intensity spinning on an exercise bike three times a week and a control group that only stretched. After 6 months, these patients were asked to complete the MDS-UPDRS score list together with the neurologist. The results showed a statistically significant difference between both groups, in favor of the intervention group. [31]

3.2.2 Deep Brain Stimulation (DBS)

3.2.2.1 Introduction

50% Of PD patients develop new complaints after 5 years of medication [32]. An example of this is dyskinesia. Often when this happens, doctors consider using DBS. DBS should ensure that motor functions continue to function as well as improvement of the memory. [32]

3.2.2.2 Continuous DBS

When applying DBS, the doctors stimulate the STN and GPi (see figure 2). They do this using electrical signals. These electrical signals reach those area's through needles, which have a contact length of 1.5 mm. The needles are then spaced 0.5 to 1.5 mm apart [4, 33]. Between 4 and 8 needles are used. The tips of these needles go to a pacemaker, which is located in the chest [33]. According to Hariz et al.[4] typical DBS uses a pulse amplitude of 1 to 4 mA and a frequency of 130Hz, with a pulse width of 60ms [4]. Introducing electrical signals to the brain areas should ensure that the overactivity of the STN or GPi decreases. As a result, the thalamo-cortical projections are less blocked. And so that at least the motor complaints decrease [4]. Once DBS has been applied, medication is still used.

3.2.2.3 STN DBS

This form of DBS is the most researched and therefore the most effective [4]. PD patients who undergo this treatment see a strong decrease in complaints and can greatly reduce their medication use [4]. The patients who receive this treatment are often relatively young. This means that they will (hopefully) get fewer complaints later on and therefore have a more optimistic future ahead. Besides, it largely resolves motor complaints [34]. It also ensures that PD patients with this form of DBS can sleep better, have a higher pain threshold and are less anxious. [4, 34]

3.2.2.4 Side effects DBS

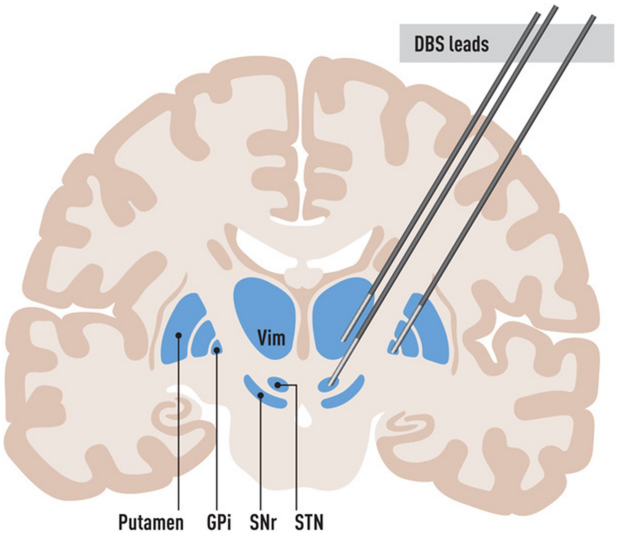

Figure 2: The working of DBS.
Note. Reprinted from "Deep brain stimulation for Parkinson's disease" by M. Hariz, P. Blomstedt, 2022, Journal of Internal medicine. Copyright 2022 The Authors. Journal of Internal Medicine published by John Wiley & Sons Ltd on behalf of Association for Publication of The Journal of Internal Medicine.[4]

Side effects can occur with all treatments, including DBS. During an operation, something can always go wrong, such as an infection. One of the most common side effects after successful surgery is dysarthria [32]. This can be caused by the fact that the STN is more active and that certain muscles are inhibited as a result. STN-DBS can cause mood swings, but can also changes the behavior of patients. Another disadvantage of DBS is that it does not dynamically adapt to the patient [32, 35]. Once programmed in, the DBS will have a continuous activation. This is independent of the patient's situation. Furthermore, the battery does not always last long [34]. This ensures that patients can unintentionally get complaints again. [4]

3.2.2.5 Adaptive DBS

With the current treatment, DBS is always on. aDBS should eventually replace this. Instead of being on all the time, like now. aDBS should only work when the patient notices that the complaints are getting worse or the patient should turn it down if he notices that he has no complaints. A normal DBS is an open loop [35, 36]. This means that adjustments have to be made from time to time. This is only possible if the DBS is switched off and patients therefore experience complaints. This is different with aDBS. This is a closed loop system [35, 36]. This means that the device adjusts automatically based on the feedback from the sensor. The closed-loop system consists of three parts. Those are the sensor, controller, and stimulator. [36]

The sensor measures whether there is a change. This change is different for brand. The different signal changes are local field potentials, electrocorticography, signals from wearables and neurochemicals [34, 36]. Any of these signals can be used. One of these signals is then forwarded to the controller. The controller

processes this signal. After it has been processed, it is calculated how much mA must be delivered and where. When the calculation is complete, the stimulator outputs the result. [34, 36]

3.3 Movement Disorder Society- Unified Parkinson's Disease Rating Scale

3.3.1 The origin of UPDRS

Due to the lack of blood tests or other imaging tools, clinicians use scales to help diagnose and measure the severity of Parkinson's Disease. The most used scale is the Movement Disorder Society- Unified Parkinson's Disease Rating Scale (MDS-UPDRS) to analytically investigate the severity of PD [37, 38].The UPDRS score list was approved in 1980, twenty-eight years later in 2008, this scale was updated and scientifically approved after the validity was researched. This study from 2008 was performed by the Movement Disorder Society giving its name to it: MDS-UPDRS [38]. To better understand the difference between those two scales, it's important to have some background information about the original scale. It consists of 55 questions divided in two categories, 7 of them are yes/no questions and the other 48 are in the form of a 5 point scale from 0 to 4. Were 0 means that the symptom is not present and a value of 4 states that the symptom is severe. Furthermore the test is divided in four parts. Part 1 focuses on non-motor symptoms; part 2 on the continuation of daily activities; part 3 is performed by a clinician and is mainly focused on symptoms of Parkinson's Disease; part 4 includes questions about motor complications. [37, 38]

With this test, clinicians can classify patients with more certainty and measure the progression of Parkinson's Disease. However, PD patients experience symptoms differently. A patient that has painting as a hobby feels more discomfort of having tremor than someone that likes to ride a bicycle [37]. This score lists gives room for different interpretations and different outcomes while the questions are the same. Another limitation is how patients memorise certain events. Some people remember the times they had trouble with the activity better than the times everything went well [37]. For example, the days they didn't succeed changing their clothes will be memorized better then the days they succeeded.

The different parts of the test are a strong aspect of the test. For the clinician it is easier to understand which symptoms have the most influence on the quality of life. If a patient is sombre and is close to a depression but experience few limitations in daily life, the physician can adjust the treatment plan to improve the mood of the patient [38]. Moreover, the comparison of each part individually gives a better understanding in which categories the patient experience worsening of symptoms and for which parts the progression of PD is limited [38].

3.3.2 MDS-UPDRS

As mentioned earlier, in 2008 the UPDRS score list was renewed. This renewal consists 65 questions: 10 more then before, of which 9 new items are not related to the original scale. These new questions were focused on both physical and mental abnormalities. The mental questions had as subjects the anxious mood and dopamine dysregulation syndrome. The physical part consists of way more questions:

urinary problems, constipation, fatigue, doing hobbies, getting in and out of bed, toe tapping, and freezing. Moreover the scale was adjusted: the UPDRS the scale goes from none (0), mild (1), moderate (2), severe (3) to marked (4). There is clinically not a relevant difference between severe and marked, and for physicians an extra category in the not-severe range has more relevance to detect and diagnose early stages of PD.

For this reason, the MDS-UPDRS starts the same at none (0) then has a new category: slight (1). In this category the activity is impacted little and with low frequency. After slight comes mild (2), in this case the impact is more clear or the symptoms occur more frequently. By moderate (3), the impact on the daily activity has to be clear and occur more frequently. The last rating severe (4) has become a combination of the previous categories (severe and marked). If the question is rated with a four, the activity is impacted heavily and there is serious function loss. [38]

Lastly, some questions are rewritten so that the patient or caregiver can answer them without a supervisor. Some questions (mostly from part 3 of the MDS-UPDRS test) still require a specialist to give a result. In total, the examination time for physicians is shorter in comparison to the original UPDRS score list. [38]

3.3.3 Limitations of MDS-UPDRS

Even with the use of MDS-UPDRS score lists, there are still some limitations to this approach. Especially how the results should be interpreted: two patients with in total the same amount of points can experience PD very different. For this reason, it is for clinicians hard to compare the disease between multiple patients[37]. In equal sizes this is even the case for individual patients, given that for part 1 (non-motor experiences of daily living) there are multiple ways to come to a total score. Part 1 consists of thirteen questions, thus the maximum amount of points one can get for this, is 52. Picture that a patient scored 26 points for this test. For one instance, the clinician will expect that the patient experiences some discomfort during daily life but not that many problems. This is mostly true if the patient rated each individual question with the value 2 (mild). However, this is not always the case, the patient can for example experience lots of problems with sleep. And no problems with cognitive disorders or obstipation. This aspect can have a big impact on the best treatment for the patient: for this patient, it will be more helpful to get sleep medication than to have laxative tablets. In the score sheet at the end of each part of the test, each question is shown separate from each other. But if the scores from separate parts are summed up, information regarding where the most problems for that patient are present is lost. [37]

3.4 Optimizing diagnostic methods

The MDS-UPDRS score lists still collect a broader range of information compared to smartwatch data. The limitation of these devices is that they can not be used to detect most non-motor symptoms such as depression or anxiety. Similarly, they only measure the motion of one arm while some patients experience more dyskinesia in their legs. These heavier forms of dyskinesia will be harder to detect. When using a smartwatch in comparison to a neurological test performed by an expert, the smartwatch has one big advantage and that is the continuous collection of data. The result is that this data gives a more detailed

view of how the motor symptoms change over time. A possibility to improve the usage of the smartwatch is that the patient can use an app to give information about how they felt during the day and at which moments they had severe symptoms.

3.5 Technology background

3.5.1 Introduction

Research regarding the technological applications on Parkinson's disease (PD) have been increasing exponentially [39]. Within this increase, there is a large focus on the motor symptoms and diagnostics of PD [39]. In addition to this, wearable smart devices ("wearables"), like smartwatches and smartphones, have seen a soar in popularity: over 40% of all articles published in regards to technology and PD, involved a wearable [39]. This continuous data stream can help the neurologist with making a diagnosis, monitoring, or adjustment of treatment [5, 39, 40]. Technology can also help the patient with the treatment of PD symptoms [41] and rehabilitation [42].

Due to advances in computing power and the internet, it is now possible for wearables to transmit their data to an artificial intelligence (AI) or a machine learning (ML) algorithm [43, 44]. This interconnectivity between devices also allows wearables to potentially influence treatments such as deep brain stimulation (DBS) or a duodopa pump [40]. In this literature study, we will explore the current state and future of wearables, and see how they interact with AI and ML algorithms and how they can help support the treatment of PD patients. In addition to this, in the methods section of this article, we will explore the study by Powers et al. [5] in particular due to our close affiliation with this study.

3.5.2 Wearables

Wearables, or wearable sensors, are worn, portable electronic devices used to measure data from a patient [45]. Because of the ubiquity of wearable technologies in the population, a lot of research is being done to see how it can be best utilised [46]. The inertial measurement unit (IMU, consisting of gyroscopes, accelerometers and magnetometers) present in almost all consumer wearables [47] enables measurement of bradykinesia [45, 46], tremors [5, 45, 46], dyskinesia [5], freezing of gait [45], sleep parameters [45] and rigidity [48]. More specialised designs of wearables have also showed promise with EEG and electromyography (EMG) measurement, using this data to analyse tremor or rapid eye movement [39]. All of this data could be used to support a diagnosis or to monitor a patient in the home setting [45].

Next to the monitoring of symptoms, this data could also be used to support treatments [39, 40, 41]. Laar et al. [41] explores the possibilities of wearables for cueing for patients with freezing of gait. Next to this, a common parameter to monitor in responses to treatment is bradykinesia and dyskinesia [39].

However, there are some shortcomings with the current wearables. Espay et al. [40] explores some caveats to watch out for with the development of wearables, like the abundance of data measured, whether or not it is relevant for the patient to measure certain data types and the discrepancies between the objective

sensor data and the subjective clinical scores. Deb et al. [39] highlights the importance to use multiple sensors, something that few studies have used.

Another shortcoming is the the lack of wearables on non-motor symptoms from PD in papers [39, 40, 49]. This means that it is difficult to truly get a complete picture of the patient from just wearables alone. Some studies have explored the possibilities on monitoring these non-motor symptoms, like orthostatic hypertension, impulsive control disorder and sleep problems. But due to the inherent difficulty of measuring these non-motor symptoms or the inability of the equipment to give an accurate measurement, it is with limited success. However, if more research is done towards the measurement of non-motor symptoms using wearables, it may show more promise than it is now. [49]

3.5.3 Machine learning

3.5.3.1 Introduction

Artificial intelligence (AI) is a field of computer science that occupies itself with intelligent algorithms [47]. Machine learning (ML) is a subset of AI, where the algorithm "learns" by reflecting the outcome of the algorithm on itself [44, 47]. ML is a relatively new field, which has seen integration with the analysis of PD since the late 2000's [50]. At first, most ML algorithms used were supervised algorithms: the data was labeled on whether or not the patient had PD or not [50, 44]. As time progressed, unsupervised algorithms (algorithms that do not need their data labeled, instead finding patterns in their data to make conclusions from [44]) became more prevalent [50].

3.5.3.2 Common Parkinson's disease-data algorithms

The algorithms that have seen the most success with IMU data from PD patients are support vector machines (SVM's), neural networks (NN's) with its derivatives, and decision trees (DT's) with its derivatives [47]. All these algorithms are supervised [44].

SVM's try to find a complex boundary between data-points, separating them in different clusters, each with their own classification, like PD or no PD [44]. To visualize, picture a map of a continent, but without country borders. A SVM would try to draw new borders for each country based on, for instance, the names and location of the towns and cities on the map, or whether or not a town belongs to a specific country.

NN's are another common algorithm used for PD data. NN's have a similar structure to neurons in biological neural networks, where the inputs (dendrites) are weighed, passed through an activation function, and that outcome is then passed onto the outputs (axons). These algorithms learn by influencing the importance (weight) of an input. Multiple of these "neurons", which are called nodes in artifical NN's, form a neural network. NN's can be made of one node, which is called a perceptron, but can also contain more nodes and more layers of nodes interconnected with each other. Because of this, neural networks tend to be quite complex, but also very versatile: many variants of NN's exist, with some common ones, like the convolutional neural network and recurrent neural networks, being popular tools to work with. [51]

DT's ask questions about the proposed data to classify it into subsets of itself. For example, if you let a DT play a game of Guess who?, it will ask itself questions to narrow down the possibilities of who it can be using a tree-like structure. The way a DT asks itself these "questions", is by checking if a variable or a combination of variables is larger than or equal to a value determined by the DT. For example, we can convert the Guess Who? question: "Does your character wear glasses?" by labeling all people without glasses with "glasses = 0", and all people with glasses with "glasses = 1" in our data set. Then, the DT can split the data set into people with glasses by checking if glasses = 1, for instance. From here on out, these statements that split the dataset into smaller subsets will be referred to as questions, for ease of reading. [52]

But how does the DT know what a "good" question is? There are multiple ways for a DT to do this. One of which is called the Gini index, which is the one that we will be using for our algorithm later on. Gini indexing checks if the data after the question is asked only contains as few classes per category as possible. To illustrate what we mean with this, picture the previous example of the Guess Who? example with glasses. The category "glasses" in this case contains only two classes: without glasses and with glasses. This splitting of data is beneficial for the DT, as its goal is to make sure that each class per category can be described as precise as possible. The Gini index is a number between 1 and 0. If the Gini index of a node is 1, than the question does not help with splitting the data at all. If the Gini index of a node is 0, the data has been split by that question in such a way, it only contains one class per category. [52]

As stated previously, a DT builds itself into a tree-like structure. The first node with a question is called the "root" of the DT, and continues up the nodes of the DT with more detailed questions. This continues until a node is made that already concludes something from the data, or no more questions can be asked to further split the data. These nodes, where no new question is given is called a "leaf" (in other words: the Gini index of this node is 0)(see figure 3). Each leaf will give the probability of what is the most likely outcome of that set of questions to get from the root to that leaf. Continuing with the Guess who? example, you can imagine that the order and the structuring of the questions matter a lot for the shape and size of the tree. For instance, if the DT starts with questions regarding the person's name, it will either be done directly (if it guessed correctly on the first try) or require a lot more questions to find the correct answer. In order to find the best possible DT, a technique called random forests (RF) can be deployed. RF's generate multiple trees and bases its outcome on the most common outcome of all trees. In order to do this, it splits the data into a random subset of itself and also limits itself to a select amount of questions per node. [52]

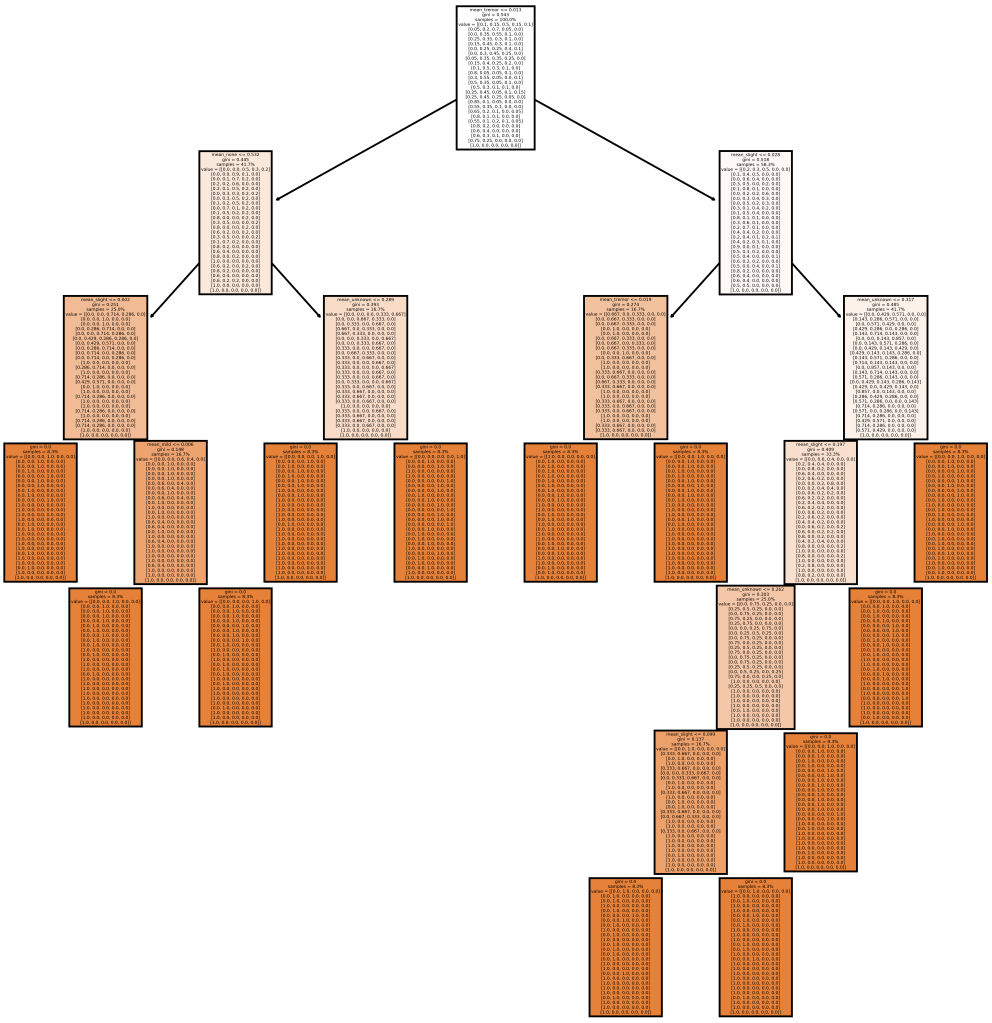

Figure 3: Example of a DT structure. The top white node in this case would be the root of the DT, and the dark, orange nodes world be the leafs.

3.5.3.3 Applications of machine learning

There are many different applications of ML in regards to PD. ML can be found in the aspects of diagnostics, the management of disease, the monitoring of disease progression and outcome prediction [53]. With regards to diagnostics, ML has found a place in analysing IMU data of the upper and lower extremities [5, 53]. Next to this, ML can also be used for analysing images, such as MRI-scans [54], or handwriting [55]. ML can possibly classify PD in different stages [56].

Monitoring and management of PD has also seen a substantial amount of publications in recent years. Many different sensor arrangements have been tried, but the most prominent one remains the IMU. Other options include: imaging and depth, audio, and pressure sensors. All of these sensors seem to have relatively the same amount of success when it comes to monitoring PD, with some senor types having more success than others. But due to the limited number of studies in some fields and with certain sensors, it is possible that this might change when more data arrives. [47]

4 Method

4.1 Study design

For our bachelor thesis we have the goal to compare MDS-UPDRS (part III) score lists with smartwatch data and find out if there is a correlation between those two. Because the smartwatch and MDS-UPDRS are quite different based on how the data is collected, we have chosen to implement an algorithm that will do the analysis.We also want to visualise the available smartwatch data by separating it in steps of ten minutes over a time span of 24 hours. This will give the physician a better understanding at which time of the day tremor or dyskinesia is more present. The second sub-goal is to compare two visits with each other based on multiple categories of the MDS-UPDRS and smartwatch data.

4.2 Selection of participants

The patients we received data from, are participants in the adaptive DBS study performed by clinicians from Amsterdam UMC. Researchers of the adaptive DBS study have selected participants based on their therapy. If they are eligible for DBS and want to participate, they will be included and receive an Apple Watch that collects the probabilities of tremor and dyskinesia.

4.3 Data collection

The smartwatch data is collected by Runelabs, an American company that is primarily focused on developing tools to improve neurological treatment. We received the MDS-UPDRS data from physicians at the Amsterdam UMC.

To perform the analysis on the smartwatch data, we first had to load these files. For us, this meant that we were able to adapt some templates offered by Runelabs to extract the data from the files. It is also possible to use the raw data for analysis, but considering the time we had for this project and the sizes (in terms of storage) of the raw data, we used pre-processed data instead.

To get a better understanding of how the pre-processing of the raw data is done, we read the Powers et al.[5] paper, as the system they developed to analyse smartwatch data from patients is very similar to our source of patient data. Mainly we are interested in the method they used to convert the raw data into probabilities. This is why we wish to discuss the strengths and limitations of this study here.

Powers et al.[5] developed a system to analyse smartwatch data to identify dyskinesia and tremors from accelerometer and gyroscope data. This system, called the Motor fluctuations Monitor for Parkinson's Disease (MM4PD), was first created and trained by analysing raw sensor data to identify episodes of dyskinesia and tremors in PD patients. Afterwards, their findings were validated by a test group.

The team determined the strength of the tremor by estimating the displacement in the frequency domain and discussed why they did this. However, some methods used patterns to determine whether tremor is present, but this method was not backed up using formulas or other scientifically based arguments.

Next to tremors, Powers et al.[5] also identified dyskinesia using the MM4PD algorithm. The team did this by first identifying when no tremor was present in 107 patients. Afterwards, using motion strength and spectral entropy as their inputs, they used a supervised ML-algorithm called "K-nearest neighbour" to use the inputs to assign data to a score. They also tested the algorithm on non-PD patients to further validate the algorithm and eliminate false-positives. Despite these flaws, everything else is explained well and thoroughly throughout the paper and supplementary materials. Powers et al.[5] explains their findings and progress in an open and sincere manner, using illustrations and video recordings where needed to illustrate their findings.

Back to our methods, we used these probabilities for our analysis and we wrote scripts to make the data accessible for others. For if other researchers or students want to use the smartwatch data later on for their own project. We think it would be helpful to write scripts that are easy in usage (partially automated), that help with storing the data in a systematic manner. After the data was extracted from the cloud and saved offline. We made a script (01) that is able to merge all the different categories (tremor probability, tremor severity and dyskinesia probability) to one Excel-file that contains all this data. Most patients have received more than one watch because of software-issues. We also made an script that is able to combine all these watches and store them in a map with the data from all other patients. This map with all the patients data together makes it very accessible for physicians to perform analyses.

For the MDS-UPDRS scorelists we got an overview containing the data from each patient for each visit. We loaded this data into Python to be able to analyse this data later on.

4.4 Data analysis

It is not possible to compare MDS-UPDRS-scores (specifically part III) with smartwatch data using statistical tests. This is because of the following reasons: firstly, the smartwatch data is collected over a period of time while the MDS-UPDRS is a crosssectional test. Secondly, both measure different types of information. Hence why we wanted to implement an algorithm to help interpret the results. For the algorithm, we used a Random Forest (RF) algorithm to give results. The reason for an RF is because the algorithm seems to perform well in many different scenarios of PD measurements and analysis [47]. To train the RF, we used both the smartwatch data and a large part of the MDS-UPDRS-scores Part III as input to train the algorithm. To test the accuracy of the system, only the smartwatch data is given and the system has to predict the corresponding MDS-UPDRS scores. The smartwatch data is processed slightly before implemented in the system. In a selected time window, around an hour in width (30 minutes before, 30 minutes after), the mean is taken from each category and used as input. The scripts also give the opportunity to visualize the data sets but this is not our main goal for this project. In order to comprehend the significance of the algorithm, statistical tests, like the Student's t-test, will be used to measure the difference between the predicted values of the RF and the test set of the RF. The advantage of using Python for performing the statistical analyses is that the library scipy.stats has plenty of different statistical functions.

As mentioned in the study design, we also set some subgoals using our pre-processed data. The first subgoal is plotting all available smartwatch data from one patient in time windows of ten minutes over a

total of 24 hours. We used the merged data from all watches that is stored in the overview map. All data points are then categorized in subsets of 10 minutes starting at night (0:00 to 0:10; 0:10 to 0:20; and so on). This plot will give the physician a better understanding of when during the day, specific symptoms are mostly present. This can be helpful for adjusting DBS or medication intake. If the physician wants to know how the most recent settings work with respect to previous settings, the physician can select a date frame of interest. For example, all data points between two visits.

5 Results

5.1 RF-algoritm and iterations

The RF-algorithm was first trained with a set of 26 MDS-UPDRS-Part-III-scores, linked to corresponding smartwatch data. We decided on using 100 estimators for this RF due to the complexity of the to be predicted data. In addition, we used 20% (n = 6) of the available data for testing, and the other 80% (n = 20) for training. The test and prediction results of one iteration can be seen in the table 1. Around half of the predicted data overlaps with the test data ("one-on-one" data), but a large amount of the data is within one increment of the correct value ("within-one" data). The accuracy was determined by the percentage of one-on-one data in the total matched data set. We measured this accuracy over 50 runs and averaged this (see appendix, table 1), to give a more global scope of the capabilities of the RF. This gives us an average accuracy of 48,9% for one-on one data and 80,9% for the within-one data. The accuracy of one-on-one data ranged from 39,1% to 63,5%, and the within-one data accuracy ranged from 71,2% up to 90,4% (see figure 3A). Using a related Student's t-test per MDS-UPDRS-score category over 50 runs of the algorithm, we concluded that most of the test and the predicted data is similar (see figure 3B and table 1). This algorithm had a sensitivity and positive predictive value of 43.6%.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
set 1	3	2	2	2	2	3	2	2	3	1	0	1	0	1	1	1	0	1	0	0	0	0	1	0	0	0
set 2	2	2	2	2	2	1	1	2	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
set 3	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
set 4	2	2	2	3	1	3	2	3	3	1	0	1	1	0	3	2	0	1	1	1	2	1	0	1	1	0
set 5	2	2	1	2	1	1	1	1	1	1	0	0	0	2	1	2	0	0	0	0	0	0	0	0	0	0
set 1	1	2	1	3	3	1	2	1	3	3	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0
set 2	1	1	1	0	0	1	1	1	2	2	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
set 3	1	2	1	0	0	1	1	1	0	3	0	1	0	3	1	1	0	0	0	0	0	0	0	0	0	0
set 4	1	1	1	0	1	3	3	1	2	2	0	1	1	1	4	0	0	1	0	0	0	0	1	0	0	0
set 5	1	1	1	0	1	1	2	1	0	2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Table 1: Test data (top) and the predictions of the first RF (bottom). Right limbs (a: finger tap, b: hand movement, c: pronation/supination, d: toe tap, e: leg movement q: postural tremor, r: kinetic tremor, s: rest tremor amplitude arm, t: rest tremor amplitude leg, u: rest tremor constancy), Left limbs (f: finger tap, g: hand movement, h: pronation/supination v: postural tremor, w: kinetic tremor, x: rest tremor amplitude arm, y: rest tremor amplitude leg, z: rest tremor constancy), i: toe tap, j: leg movement, k: raise chair, l: gait, m: freezing, n: postural reflex, o: posture, p: spontaneous movements.

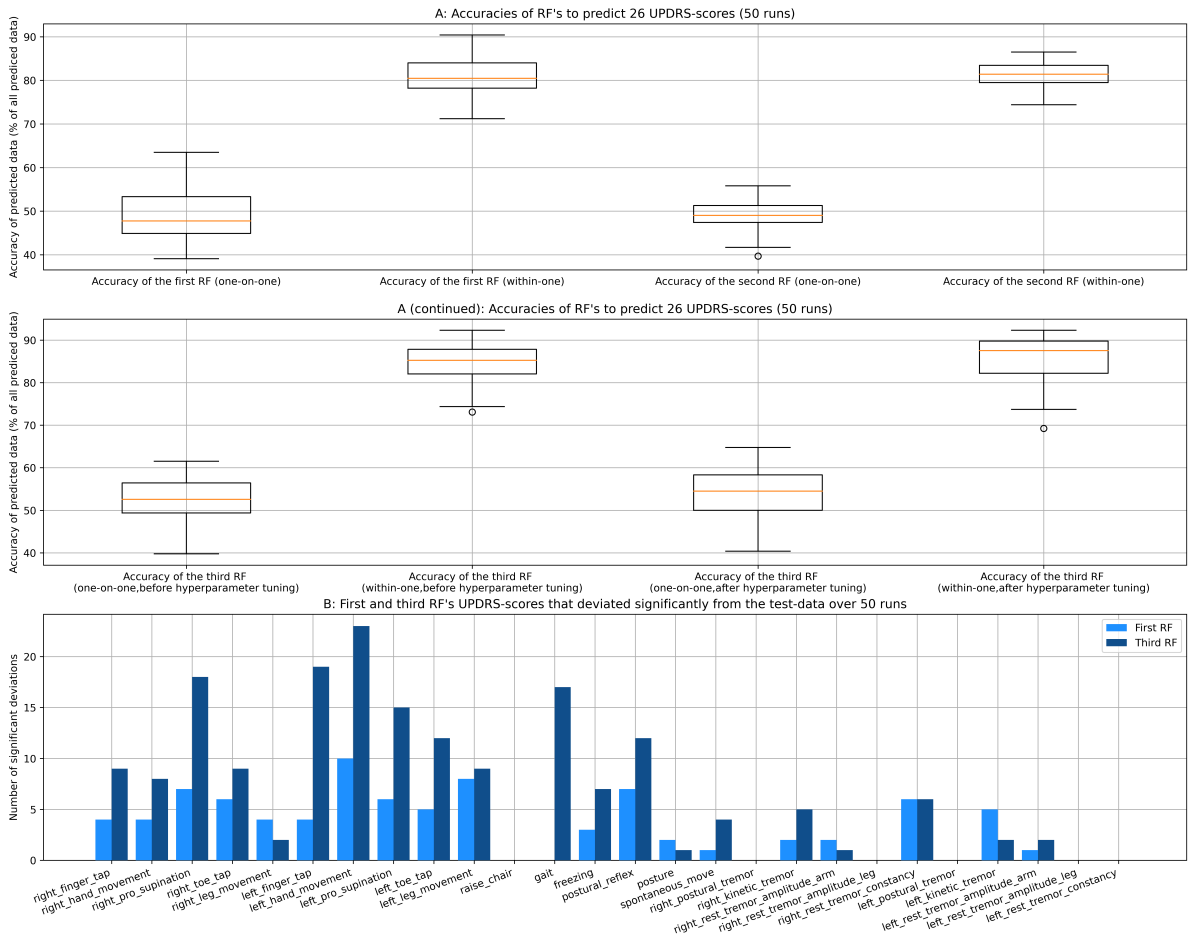

Figure 4: A: The one-on-one and within-one accuracies of the first and second RF. A(continued): The one-on-one and within-one accuracies of the third RF, before and after adjusting the hyperparameter tuning. B: The number of significant deviations per category that the first and third RF try to predict. As can be seen, over 50 runs, the third RF seems to get almost double the significant deviations, yet it maintains a higher overall accuracy. In both algorithms,the left-hand-movement UPDRS-score was the most common outlier.

With the same set of MDS-UPDRS-Part-III-scores and smartwatch data, we decided to train an RF for each category of the MDS-UPDRS-Part-III-score to be predicted. This amounted to 26 RF's, each with 100 estimators. Again, a 20%/80% test/train split was done to fit the RF-models. Each RF was run 50 times to test the accuracy of the algorithm. Results for test and prediction results can be seen in the table below. The average accuracy is 49,0% for the one-on-one category and 81,6% for the within-one category. Accuracy ranges ranged from 39,7% up to 55,8% for the one-on-one measurements, and the within-one accuracies ranged from 74,4% up to 86,5% (see figure 3A). The 26 separate algorithms show a narrower spread of the accuracies over these 50 runs when compared to the one RF (see figure 4A). However, there seems to be no increase in overall accuracy, which is what we hoped for.

Because both of these algorithms got lackluster results, we decided to try and increase the accuracy by using a process called "hyperparameter tuning" to find the best values for parameters used in the RF. This process loops through all possible combinations of variables that you enter for the parameters, picks

out the best value for that parameter, and uses it to fit the RF. We used hyperparameter on the following parameters of the RF: Number of estimators (the number of DT's per RF), the number of samples needed for a split and to determine a leaf node, bootstrapping (whether or not the RF should use all samples to train per RF), the maximum depth of the RF and if the RF should warm-start (to re-use settings of a previous RF). This process gives an one-on-one accuracy of 52,7% and a within-one accuracy of 84,8%, but training times for a set of 50 RF's went from mere seconds to around half a hour (see figure and Appendix: table 3). The one-on-one accuracy ranged from 39,7% up to 61,5% and the within-one accuracy ranged from 73,1% up to 92,3% (see figure 4). The accuracies seen in these plots are higher than that of the other algorithms, but the performance is lackluster: it takes up to 360x longer to compute 50 RF's of the new algorithm than it takes to compute 50 RF's of the old algorithm.

Although the third RF obtained higher overall accuracies, when checking for significant deviations, it shows that the third RF gets more than double the significant deviations than the first RF (see figure 4B). This means that the third RF places more emphasis on certain variables, whilst leaving others not accounted for. In order to maximize accuracy without significant increases in time, we have looked at the parameters that gave the highest accuracy in the hyperparameter tuning, and limit the tuning to a smaller margin around those values. In addition, we will enable the RF to use multiple cores from the central processing unit (CPU) in your PC. These decisions made the RF do 50 runs in about 10 minutes, whilst not lowering the accuracy of the RF. This was about the limit of what could be produced from the RF, without requiring more data.

5.2 Smartwatch data per ten minutes

After the pre-processing of the data was done and the algorithm was working. We set some sub-goals to further perform analyses on the available smartwatch data. In figure 5 we plotted the mean tremor and dyskinesia for each 10 minutes over a time span of 24 hours to get a better understanding of which symptoms are more present in the morning and which occur more often in the afternoon. The input data was the probability data, from all Apple Watches worn by the patient and separated and categorized in time windows of 10 minutes. Below this graphic is shown how much data points are available for each time window. This tells something about the preciseness of the result. If there are only a few data points available the impact of outliers increases.

5.3 Comparison of visits for MDS-UPDRS and smartwatch data

To get a better understanding of the differences in symptoms between two visits. It can be helpful to have a visual overview of how parts of the MDS-UPDRS have different scores between two visits. More interestingly is the change between visit two and three because between these two visits the DBS has been activated. The results of the MDS-UPDRS can be matched with the smartwatch data at the moment the MDS-UPDRS test was taken. For this case, we chose a time window of fifteen minutes prior and sixty minutes after MDS-UPDRS. As shown in figure 6 both the mean tremor from the Apple Watch as the scores from the tremor category of the MDS-UPDRS dropped.

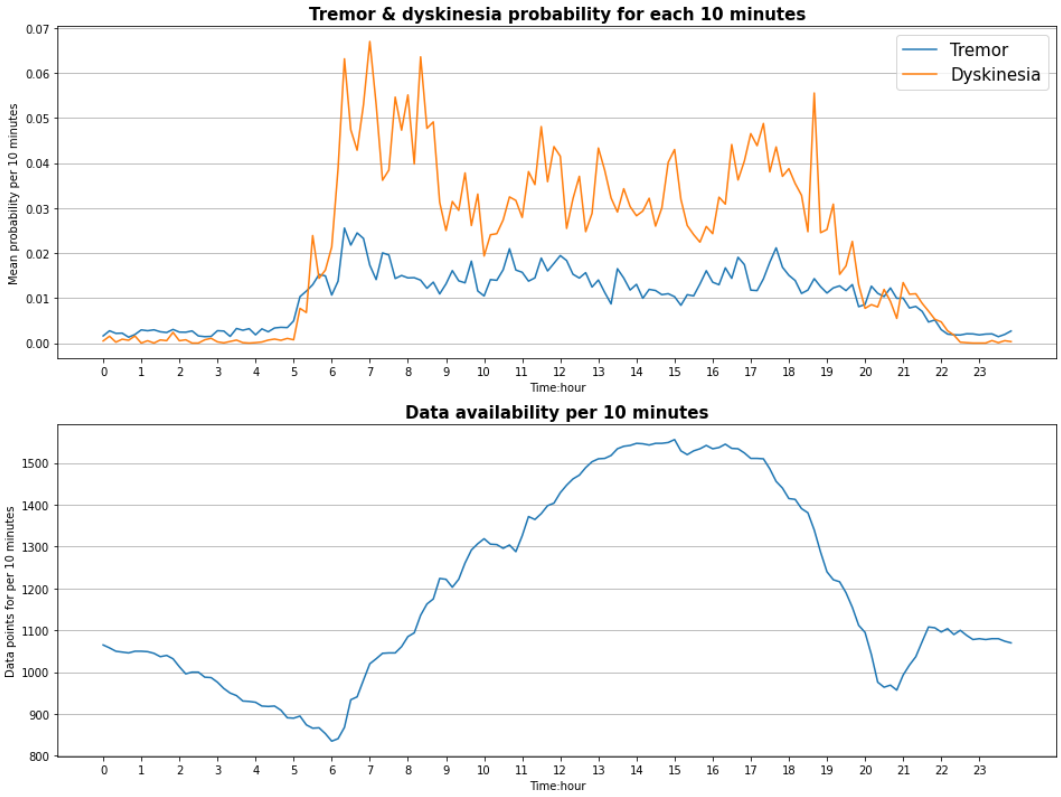

Figure 5: This figure shows the tremor and dyskinesia for each hour based on all probability data collected by the Apple Watch. The second image shows the available data points for each hour given that less data points mean that the results will be less precise

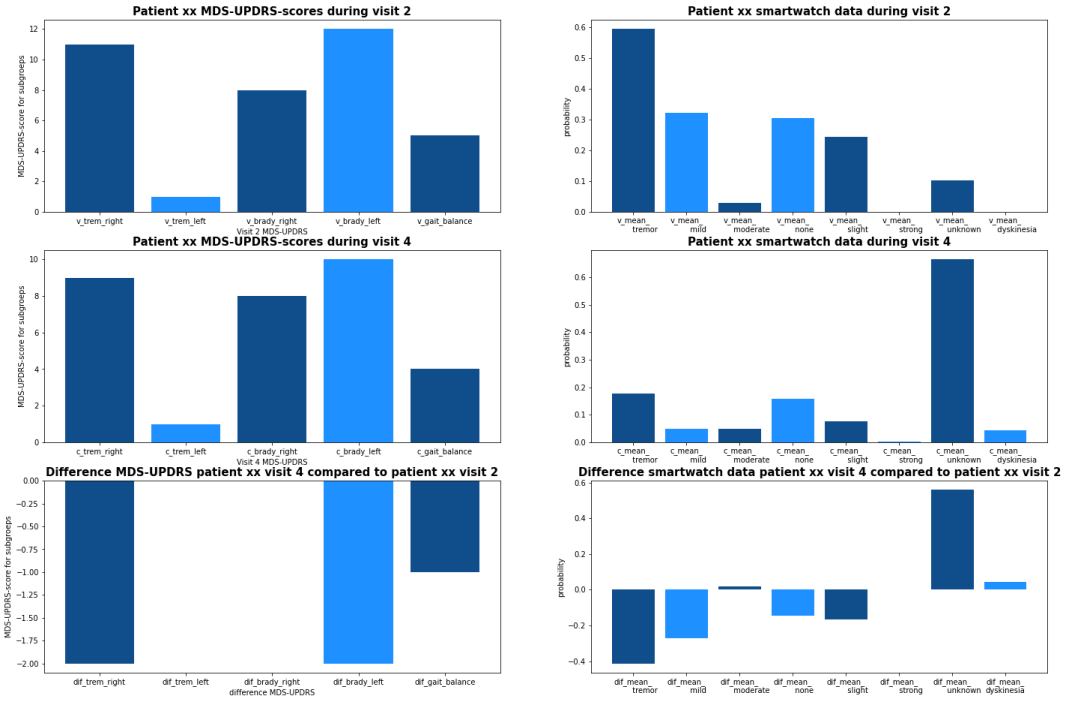

Figure 6: Comparison of the MDS-UPDRS and smartwatch data between two visits.

6 Discussion

6.1 Literature

In the theoretical background their is less focus on the specific symptoms of how they arise. This has been done because the researchers wanted to focus on the bigger picture. Instead of focusing on the specific details of symptoms. In addition, a small piece has been written about the non-motor complaints. It is important to realize that these complaints are certainly no less important and, just like motor complaints, can have a major impact on someone's life. But due to the focus on motor complaints, this is less extensive. However, the two points mentioned above are interesting for possible follow-up studies. How can the non-motor complaints be measured and how do the specific symptoms of a patient relate to the anatomy? Any other research could be aimed at the further use of AI, ML and DL aimed at PD.

6.2 Data & coding

Data acquisition was relatively difficult due to infrequent measures of the MDS-UPDRS-scorelist. Because of this, it was at times difficult to find data that matched the measurement date ("visit") of the MDS-UPDRS-score. This had us resort to either omitting the data from the ML-algorithm or using data that was (at most) hours away from the visit. This could mean that the ML has been trained with improper data, which may give selection bias on the performance of the RF.

For our algorithm we used two different time windows around the visits. The time of visit was based on the moment the neural data was read out. This read-out can only be done in the hospital. The first window was –15 minutes and + 60 minutes compared to the read-out time. The second window was –45 minutes and + 15 minutes. There was no significant difference between the time windows for the accuracy of the algorithm 52.7% accuracy for the first window and 51.4% for the second window.

Next to this, the MDS-UPDRS-scorelist was, for some patients and visits, irretrievable due to improper syntax in the documentation. As it was very difficult to compensate for this directly, it was decided to not include visits at which this improper documentation took place. As this improper documentation mainly happened to MDS-UPDRS-score that amounted to 0 (in other words: little to no motor symptoms of PD), the ML-algorithm could be biased to patients that have little to none motor hinder from PD.

Another thing of note was a lack of variability in the submitted MDS-UPDRS-scores. categories, such as the tremor scores for both right and left, were often filled with the same value, commonly 0. This means that there was little data to train with when these categories did have a value above 0. This means that there is a high likelihood that the RF has a difficulty identifying tremors. Other categories like these include: rising from chair, freezing, and gait.

Besides these negative points, there were also positive points. The various written scripts are almost all automated, only patient number, patient ID and smartwatch number need to be changed. Specifically, processing large amounts of data is fully automatic, such as loading MDS-UPDRS scores and bringing both smartwatch data and MDS-UPDRS into an Excel-file. This ensures that people who are not specialized

in programming can also retrieve and analyze the data. In addition, the scripts also display graphs. Visualization can help people enormously to process things. The graph can also help in choosing the moment to administer medication or to adjust aDBS. The time interval can be chosen by the person himself. This makes it possible to compare current data with data from months ago. It is therefore possible to see whether progress has been made or whether there has been a regression. And therefore see if anything needs to be changed in the current treatment plan.

Any further studies could focus on processing other major impacting symptoms and comparing them to what is now available. But also look at how you can take the data from smartwatches at the same time as the MDS-UPDRS scores and how to store this data. So that this data can be found more easily. A next step could also be to create a (better) interface. This should make it even easier for patient and doctor to analyse the data.

6.3 ML-performance

Considering the limited data availability and the inherent subjectiveness of the MDS-UPDRS-score data that we did have, and in some cases the limited data variability, we think the RF worked well. Especially the third RF-algorithm gave quite promising results with the available data. We do think the RF can be improved by using additional, and more varied training- and testing data. This way, the RF can explore more scenario's which will give less outliers and thus give a more accurate outcome.

In addition to this, exploring more types of ML may lead to one that gives better accuracy with the given data. Lastly, reusing the RF to train from the accellerometer- and gyroscope data instead of the probability of tremor, tremor severity and dyskinesia may also give more reliable results due to the objectiveness of that data.

The big advantage that ML brings is that the data is compared by an independent person/thing. Provided that it is well trained. It has little to no bias and is less time consuming than if one person tried to analyse it. Currently, our ML gives a higher predictive value for an MDS-UPDRS score than if a person were to gamble. This is an important point when one realizes that our data only had 26 patients.

Interesting follow-up studies focusing on ML may involve using other forms of ML to analyse this data. Other studies may be aimed at improving the current ML type. The use of ML in combination with AI and DL could also be considered.

6.4 Implication for patient

Patients participating in our part of the Amsterdam UMC study experience no extra burden. After all the smart watch data is already automatically collected and the MDS-UPDRS assessments are part of the standard care. The goal of this study is to give physicians and patients a better overview of when symptoms are mostly present. With this overview the treatment for each patient can be adjusted based on their current status. For now, the algorithm is not precise enough to give meaningful predictions. If the algorithm is properly trained, it can give meaningful predictions of the disease.

7 Conclusion

In the conclusion, we answer our main question. It reads as follows: What is the correlation between outcome measures of the smartwatch data and MDS-UPDRS scores of patients with Parkinson's disease?

To answer our sub-questions, we did a thorough literature study regarding PD and ML, which can be read in the "Theoretical background" section. Furthermore, to answer the technical sub-questions, we made multiple scripts to read, filter and write data so it can be used by the RF to be analysed. For more details and the exact content of the scripts used, we refer you to the appendix. Here, a "ReadMe" is given about the scripts used in this research. Next to this, the "Results" section writes about the end products of these scripts.

With the current data available, it is not possible to classify or correlate parts of the MDS-UPDRS part-III scorelist accurately with the smartwatch data. Even though the RF algorithm can get percentages around 86% for scores within one of the correct value, we do not think this is large enough to be able to justify a correlation between the MDS-UPDRS Part-III scorelist, and the smartwatch data measured by Apple watches.

References

[1] Tolosa E, Garrido A, Scholz SW, Poewe W. Challenges in the diagnosis of Parkinson's disease. The Lancet Neurology. 2021 5;20(5):385. Available from: /pmc/articles/PMC8185633//pmc/articles/PMC8185633/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC8185633/. doi:10.1016/S1474-4422(21)00030-2.

[2] Lanciego JL, Luquin N, Obeso JA. Functional neuroanatomy of the basal ganglia. Cold Spring Harbor Perspectives in Medicine. 2012 12;2(12):a009621–a009621. doi:10.1101/cshperspect.a009621.

[3] Fazl A, Fleisher J. Anatomy, Physiology, and Clinical Syndromes of the Basal Ganglia: A Brief Review. Seminars in pediatric neurology. 2018 4;25:2. Available from: /pmc/articles/PMC6039104//pmc/articles/PMC6039104/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC6039104/. doi:10.1016/J.SPEN.2017.12.005.

[4] Hariz M, Blomstedt P. Deep brain stimulation for Parkinson's disease. Journal of Internal Medicine. 2022 11;292(5):764–778. Available from: https://onlinelibrary.wiley.com/doi/10.1111/joim.13541. doi:10.1111/joim.13541.

[5] Powers R, Etezadi-Amoli M, Arnold EM, Kianian S, Mance I, Gibiansky M, et al. Smartwatch inertial sensors continuously monitor real-world motor fluctuations in Parkinson's disease. Science Translational Medicine. 2021 2;13(579). Available from: https://www.science.org/doi/10.1126/scitranslmed.abd7865. doi:10.1126/scitranslmed.abd7865.

[6] Chouinard PA, Paus T. The Primary Motor and Premotor Areas of the Human Cerebral Cortex; 2006. doi:10.1177/1073858405284255.

[7] Purves D, Augustine GJ, Fitzpatrick D, Katz LC, LaMantia AS, McNamara JO, et al. The Premotor Cortex. vol. 2nd editio; 2001. Available from: https://www.ncbi.nlm.nih.gov/books/NBK10796/.

[8] Ninomiya T, Inoue Ki, Hoshi E, Takada M. Layer specificity of inputs from supplementary motor area and dorsal premotor cortex to primary motor cortex in macaque monkeys. Scientific Reports. 2019 12;9(1):18230. doi:10.1038/s41598-019-54220-z.

[9] Ackerman S. Discovering the Brain. Washington, D.C.: National Academies Press; 1992. Available from: http://www.nap.edu/catalog/1785. doi:10.17226/1785.

[10] Bingham CS, Petersen MV, Parent M, McIntyre CC. Evolving characterization of the human hyperdirect pathway. Brain Structure and Function. 2023 3;228(2):353–365. Available from: https://link.springer.com/article/10.1007/s00429-023-02610-5. doi:10.1007/S00429-023-02610-5/FIGURES/8.

[11] Herrington TM, Cheng JJ, Eskandar EN. Mechanisms of deep brain stimulation. Journal of Neurophysiology. 2016 1;115(1):19–38. Available from: https://www.physiology.org/doi/10.1152/jn.00281.2015. doi:10.1152/jn.00281.2015.

[12] Lashuel HA, Overk CR, Oueslati A, Masliah E. The many faces of α-synuclein: from structure and toxicity to therapeutic target. Nature reviews Neuroscience. 2013 1;14(1):38. Available from: /pmc/articles/PMC4295774//pmc/articles/PMC4295774/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC4295774/. doi:10.1038/NRN3406.

[13] Marogianni C, Sokratous M, Dardiotis E, Hadjigeorgiou GM, Bogdanos D, Xiromerisiou G. Neurodegeneration and Inflammation—An Interesting Interplay in Parkinson's Disease. International Journal of Molecular Sciences. 2020 11;21(22):1–15. Available from: /pmc/articles/PMC7697354//pmc/articles/PMC7697354/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC7697354/. doi:10.3390/IJMS21228421.

[14] Han Y, Wu D, Wang Y, Xie J, Zhang Z. Skin alpha-synuclein deposit patterns: A predictor of Parkinson's disease subtypes. eBioMedicine. 2022 6;80. Available from: /pmc/articles/PMC9148991//pmc/articles/PMC9148991/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC9148991/. doi:10.1016/J.EBIOM.2022.104076.

[15] Koziol LF, Budding D, Andreasen N, D'Arrigo S, Bulgheroni S, Imamizu H, et al.. Consensus paper: The cerebellum's role in movement and cognition; 2014. doi:10.1007/s12311-013-0511-x.

[16] Proville RD, Spolidoro M, Guyon N, Dugué GP, Selimi F, Isope P, et al. Cerebellum involvement in cortical sensorimotor circuits for the control of voluntary movements. Nature Neuroscience. 2014 9;17(9):1233–1239. doi:10.1038/nn.3773.

[17] Jamwal S, Kumar P. Insight Into the Emerging Role of Striatal Neurotransmitters in the Pathophysiology of Parkinson's Disease and Huntington's Disease: A Review. Current Neuropharmacology. 2018 1;17(2):165–175. doi:10.2174/1570159x16666180302115032.

[18] Ford CP. The role of D2-autoreceptors in regulating dopamine neuron activity and transmission. Neuroscience. 2014 12;282:13–22. Available from: http://www.ncbi.nlm.nih.gov/pubmed/24463000. doi:10.1016/j.neuroscience.2014.01.025.

[19] Roohi-Azizi M, Azimi L, Heysieattalab S, Aamidfar M. Changes of the brain's bioelectrical activity in cognition, consciousness, and some mental disorders. Medical journal of the Islamic Republic of Iran. 2017;31:53. doi:10.14196/mjiri.31.53.

[20] Florin E, Erasmi R, Reck C, Maarouf M, Schnitzler A, Fink GR, et al. Does increased gamma activity in patients suffering from Parkinson's disease counteract the movement inhibiting beta activity? Neuroscience. 2013 5;237:42–50. doi:10.1016/j.neuroscience.2013.01.051.

[21] Sonne J, Reddy V, Beato MR. Neuroanatomy, Substantia Nigra. StatPearls Publishing; 2023. Available from: https://pubmed.ncbi.nlm.nih.gov/30725680/https://www.ncbi.nlm.nih.gov/books/NBK536995/.

[22] Bourdy R, Sánchez-Catalán MJ, Kaufling J, Balcita-Pedicino JJ, Freund-Mercier MJ, Veinante P, et al. Control of the nigrostriatal dopamine neuron activity and motor function by the tail of the ventral tegmental area. Neuropsychopharmacology. 2014 11;39(12):2788–2798. doi:10.1038/npp.2014.129.

[23] Roos DS, Klein M, Deeg DJH, Doty RL, Berendse HW. Prevalence of Prodromal Symptoms of Parkinson's Disease in the Late Middle-Aged Population. Journal of Parkinson's Disease. 2022 4;12(3):967–974. doi:10.3233/JPD-213007.

[24] Bloem BR, Okun MS, Klein C. Parkinson's disease. The Lancet. 2021 6;397(10291):2284–2303. Available from: https://linkinghub.elsevier.com/retrieve/pii/S014067362100218X. doi:10.1016/S0140-6736(21)00218-X.

[25] Martínez-Martín P, Rodríguez-Blázquez C, Alvarez M, Arakaki T, Arillo VC, Chaná P, et al. Parkinson's disease severity levels and MDS-Unified Parkinson's Disease Rating Scale. Parkinsonism and Related Disorders. 2015 1;21(1):50–54. Available from: https://linkinghub.elsevier.com/retrieve/pii/S1353802014004118. doi:10.1016/j.parkreldis.2014.10.026.

[26] Neff C, Wang MC, Martel H. Using the PDQ-39 in routine care for Parkinson's disease. Parkinsonism and Related Disorders. 2018 8;53:105–107. Available from: https://linkinghub.elsevier.com/retrieve/pii/S1353802018302517. doi:10.1016/j.parkreldis.2018.05.019.

[27] Emamzadeh FN, Surguchov A. Parkinson's Disease: Biomarkers, Treatment, and Risk Factors. Frontiers in Neuroscience. 2018 8;12. doi:10.3389/fnins.2018.00612.

[28] Reich SG, Savitt JM. Parkinson's Disease. Medical Clinics of North America. 2019 3;103(2):337–350. doi:10.1016/J.MCNA.2018.10.014.

[29] Wecker L, Taylor DA. Brody ' s Human Pharmacology. 6th ed. Elsevier; 2018.

[30] Gandhi KR, Saadabadi A. Levodopa (L-Dopa). Treasure Island (FL): StatPearls Publishing; 2023. Available from: https://pubmed.ncbi.nlm.nih.gov/29489269/https://www.ncbi.nlm.nih.gov/books/NBK482140/. doi:10.4324/9781315825748-35.

[31] van der Kolk NM, de Vries NM, Kessels RPC, Joosten H, Zwinderman AH, Post B, et al. Effectiveness of home-based and remotely supervised aerobic exercise in Parkinson's disease: a double-blind, randomised controlled trial. The Lancet Neurology. 2019 11;18(11):998–1008. doi:10.1016/S1474-4422(19)30285-6.

[32] Shah H, Usman O, Ur Rehman H, Jhaveri S, Avanthika C, Hussain K, et al. Deep Brain Stimulation in the Treatment of Parkinson's Disease. Cureus. 2022 9. Available from: https://www.cureus.com/articles/106888-deep-brain-stimulation-in-the-treatment-of-parkinsons-disease. doi:10.7759/cureus.28760.

[33] Malvea A, Babaei F, Boulay C, Sachs A, Park J. Deep brain stimulation for Parkinson's Disease: A Review and Future Outlook. Biomedical Engineering Letters. 2022 8;12(3):303–316. Available from: https://link.springer.com/10.1007/s13534-022-00226-y. doi:10.1007/s13534-022-00226-y.

[34] Habets JGV, Heijmans M, Kuijf ML, Janssen MLF, Temel Y, Kubben PL. An update on adaptive deep brain stimulation in Parkinson's disease. Movement Disorders. 2018 12;33(12):1834–1843. Available from: https://onlinelibrary.wiley.com/doi/10.1002/mds.115. doi:10.1002/mds.115.

[35] Swann NC, De Hemptinne C, Thompson MC, Miocinovic S, Miller AM, Gilron R, et al. Adaptive deep brain stimulation for Parkinson's disease using motor cortex sensing. Journal of Neural Engineering. 2018 8;15(4):046006. Available from: https://iopscience.iop.org/article/10.1088/1741-2552/aabc9b. doi:10.1088/1741-2552/aabc9b.

[36] Guidetti M, Marceglia S, Loh A, Harmsen IE, Meoni S, Foffani G, et al. Clinical perspectives of adaptive deep brain stimulation. Brain Stimulation. 2021 9;14(5):1238–1247. Available from: https://linkinghub.elsevier.com/retrieve/pii/S1935861X21002023. doi:10.1016/j.brs.2021.07.063.

[37] Hendricks RM, Khasawneh MT. An Investigation into the Use and Meaning of Parkinson's Disease Clinical Scale Scores. Parkinson's Disease. 2021 5;2021:1–7. doi:10.1155/2021/1765220.

[38] Goetz CG, Tilley BC, Shaftman SR, Stebbins GT, Fahn S, Martinez-Martin P, et al. Movement Disorder Society-sponsored revision of the Unified Parkinson's Disease Rating Scale (MDS-UPDRS): Scale presentation and clinimetric testing results. Movement Disorders. 2008 11;23(15):2129–2170. doi:10.1002/mds.22340.

[39] Deb R, An S, Bhat G, Shill H, Ogras UY. A Systematic Survey of Research Trends in Technology Usage for Parkinson's Disease. Sensors. 2022 7;22(15):5491. Available from: https://www.mdpi.com/1424-8220/22/15/5491. doi:10.3390/s22155491.

[40] Espay AJ, Bonato P, Nahab FB, Maetzler W, Dean JM, Klucken J, et al. Technology in Parkinson's disease: Challenges and opportunities. Movement Disorders. 2016 9;31(9):1272–1282. Available from: https://onlinelibrary.wiley.com/doi/10.1002/mds.26642. doi:10.1002/mds.26642.

[41] Laar A, Silva de Lima AL, Maas BR, Bloem BR, de Vries NM. Successful implementation of technology in the management of Parkinson's disease: Barriers and facilitators. Clinical Parkinsonism and Related Disorders. 2023;8:100188. Available from: https://linkinghub.elsevier.com/retrieve/pii/S2590112523000063. doi:10.1016/j.prdoa.2023.100188.

[42] Lau J, Regis C, Burke C, Kaleda MJ, McKenna R, Muratori LM. Immersive Technology for Cognitive-Motor Training in Parkinson's Disease. Frontiers in Human Neuroscience. 2022 5;16. Available from: https://www.frontiersin.org/articles/10.3389/fnhum.2022.863930/full. doi:10.3389/fnhum.2022.863930.

[43] Godoi BB, Amorim GD, Quiroga DG, Holanda VM, Júlio T, Tournier MB. Parkinson's disease and wearable devices, new perspectives for a public health issue: an integrative literature review. Revista da Associacao Medica Brasileira (1992). 2019 11;65(11):1413–1420. Available from: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-42302019001101413&tlng=en. doi:10.1590/1806-9282.65.11.1413.

[44] Kubota KJ, Chen JA, Little MA. Machine learning for large-scale wearable sensor data in Parkinson's disease: Concepts, promises, pitfalls, and futures. Movement Disorders. 2016 9;31(9):1314–1326. Available from: https://movementdisorders-onlinelibrary-wiley-com.ezproxy2.utwente.nl/doi/full/10.1002/mds.26693?sid=worldcat.org. doi:10.1002/mds.26693.

[45] Lu R, Xu Y, Li X, Fan Y, Zeng W, Tan Y, et al. Evaluation of Wearable Sensor Devices in Parkinson's Disease: A Review of Current Status and Future Prospects. Parkinson's Disease. 2020 9;2020:1–8. Available from: https://www.hindawi.com/journals/pd/2020/4693019/. doi:10.1155/2020/4693019.

[46] Sánchez-Ferro Elshehabi M, Godinho C, Salkovic D, Hobert MA, Domingos J, et al. New methods for the assessment of Parkinson's disease (2005 to 2015): A systematic review. Movement Disorders. 2016 9;31(9):1283–1292. Available from: https://onlinelibrary.wiley.com/doi/10.1002/mds.26723. doi:10.1002/mds.26723.

[47] Giannakopoulou KM, Roussaki I, Demestichas K. Internet of Things Technologies and Machine Learning Methods for Parkinson's Disease Diagnosis, Monitoring and Management: A Systematic Review. Sensors. 2022 2;22(5):1799. Available from: https://www.mdpi.com/1424-8220/22/5/1799. doi:10.3390/s22051799.

[48] Wu Z, Jiang X, Zhong M, Shen B, Zhu J, Pan Y, et al. Wearable Sensors Measure Ankle Joint Changes of Patients with Parkinson's Disease before and after Acute Levodopa Challenge. Parkinson's Disease. 2020 4;2020:1–7. Available from: https://www.hindawi.com/journals/pd/2020/2976535/. doi:10.1155/2020/2976535.

[49] van Wamelen DJ, Sringean J, Trivedi D, Carroll CB, Schrag AE, Odin P, et al. Digital health technology for non-motor symptoms in people with Parkinson's disease: Futile or future? Parkinsonism & Related Disorders. 2021 8;89:186–194. Available from: https://linkinghub.elsevier.com/retrieve/pii/S1353802021002819. doi:10.1016/j.parkreldis.2021.07.032.

[50] Chandrabhatla AS, Pomeraniec IJ, Ksendzovsky A. Co-evolution of machine learning and digital technologies to improve monitoring of Parkinson's disease motor symptoms. npj Digital Medicine. 2022 3;5(1):32. Available from: https://www.nature.com/articles/s41746-022-00568-y. doi:10.1038/s41746-022-00568-y.

[51] Aggarwal CC. Neural Networks and Deep Learning. Cham: Springer International Publishing; 2018. Available from: http://link.springer.com/10.1007/978-3-319-94463-0. doi:10.1007/978-3-319-94463-0.

[52] Kingsford C, Salzberg SL. What are decision trees? Nature Biotechnology. 2008 9;26(9):1011–1012. Available from: http://www.nature.com/articles/nbt0908-1011. doi:10.1038/nbt0908-1011.

[53] Belić M, Bobić V, Badža M, Šolaja N, Đurić Jovičić M, Kostić VS. Artificial intelligence for assisting diagnostics and assessment of Parkinson's disease—A review. Clinical Neurology and Neurosurgery. 2019 9;184:105442. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0303846719302380. doi:10.1016/j.clineuro.2019.105442.

[54] Xu J, Zhang M. Use of Magnetic Resonance Imaging and Artificial Intelligence in Studies of Diagnosis of Parkinson's Disease. ACS Chemical Neuroscience. 2019 6;10(6):2658–2667. Available from: https://pubs.acs.org/doi/10.1021/acschemneuro.9b00207. doi:10.1021/acschemneuro.9b00207.

[55] Rana A, Dumka A, Singh R, Panda MK, Priyadarshi N, Twala B. Imperative Role of Machine Learning Algorithm for Detection of Parkinson's Disease: Review, Challenges and Recommendations. Diagnostics. 2022 8;12(8):2003. Available from: https://www.mdpi.com/2075-4418/12/8/2003. doi:10.3390/diagnostics12082003.

[56] Templeton JM, Poellabauer C, Schneider S. Classification of Parkinson's disease and its stages using machine learning. Scientific Reports. 2022 8;12(1):14036. Available from: https://www.nature.com/articles/s41598-022-18015-z. doi:10.1038/s41598-022-18015-z.

8 Appendix

8.1 Tables

Table 1: Accuracies of 1 RF to predict UPDRS

Run	Accuracy (one-on-one)(%)	Accuracy (Within-one)(%)	Run	Accuracy (one-on-one)(%)	Accuracy (Within-one)(%)
1	56,4	80,8	26	55,8	80,8
2	51,9	84,0	27	48,7	78,2
3	61,5	87,2	28	42,9	78,8
4	62,8	87,8	29	50,0	84,0
5	57,1	83,3	30	49,4	75,0
6	44,9	78,8	31	42,9	71,2
7	41,9	76,3	32	48,1	85,3
8	50,6	84,0	33	44,9	73,1
9	53,8	84,0	34	39,7	74,4
10	45,5	80,8	35	46,2	82,1
11	48,1	82,7	36	57,1	85,3
12	45,5	76,9	37	51,9	86,5
13	43,6	76,3	38	53,8	85,3
14	45,5	79,5	39	47,4	78,2
15	46,2	80,1	40	39,1	73,7
16	44,2	76,3	41	54,5	86,5
17	41,0	79,5	42	41,0	79,5
18	55,1	84,0	43	48,7	80,8
19	43,6	80,1	44	63,5	90,4
20	46,8	78,2	45	48,1	86,5
21	41,0	73,1	46	60,9	87,2
22	46,8	79,5	47	48,1	78,8
23	47,4	78,8	48	45,5	80,8
24	57,7	89,7	49	41,0	78,2
25	50,6	82,7	50	45,5	77,6

Table 2: Accuracies of 26 RF's to predict sepate UPDRS elements					
Run	Accuracy (one-on-one)(%)	Accuracy (Within-one)(%)	Run	Accuracy (one-on-one)(%)	Accuracy (Within-one)(%)
1	48,7	79,5	26	49,4	80,8
2	53,2	82,7	27	51,9	80,8
3	51,3	78,8	28	51,3	86,5
4	53,2	80,1	29	52,6	79,5
5	47,4	79,5	30	51,9	83,5
6	51,3	80,8	31	49,4	86,5
7	53,8	84,0	32	50,0	84,0
8	48,7	78,8	33	55,8	84,6
9	50,0	82,7	34	54,5	84,6
10	55,1	84,6	35	39,7	74,4
11	46,2	82,1	36	47,4	80,8
12	46,2	82,1	37	43,6	76,9
13	51,3	82,1	38	41,7	77,6
14	52,6	85,9	39	54,5	84,6
15	50,0	80,8	40	46,1	76,3
16	50,6	81,4	41	48,7	82,7
17	50,0	85,3	42	48,1	79,5
18	50,0	86,5	43	47,4	82,7
19	45,5	81,4	44	50,6	84,6
20	48,1	80,1	45	52,6	85,3
21	42,9	78,2	46	48,7	83,3
22	47,4	78,4	47	42,3	79,5
23	48,7	79,5	48	49,4	80,8
24	47,4	81,4	49	47,4	82,1
25	44,9	82,1	50	48,1	81,4

Table 3: Accuracies of the third algorithm's RF to predict seperate UPDRS values					
Run	Accuracy (one-on-one)(%)	Accuracy (Within-one)(%)	Run	Accuracy (one-on-one)(%)	Accuracy (Within-one)(%)
1	51,9	84,6	26	51,3	79,5
2	60,9	87,8	27	51,9	87,8
3	56,4	90,4	28	48,1	78,8
4	44,9	75,0	29	52,6	85,9
5	57,1	89,1	30	60,9	89,1
6	51,9	85,3	31	46,2	86,5
7	39,7	74,4	32	55,8	88,5
8	55,8	91,0	33	45,5	77,6
9	52,6	84,6	34	55,8	87,8
10	55,1	85,3	35	50,6	84,0
11	53,8	84,0	36	53,8	84,6
12	59,6	91,7	37	45,5	73,1
13	50,6	83,3	38	56,4	80,1
14	41,7	77,6	39	48,1	82,1
15	53,2	85,9	40	53,2	85,9
16	52,6	85,9	41	58,3	86,5
17	49,4	85,3	42	46,8	85,3
18	60,9	85,9	43	59,0	87,2
19	50,6	82,1	44	56,4	91,0
20	51,3	82,7	45	57,7	88,5
21	52,6	88,5	46	47,4	84,6
22	49,4	81,4	47	45,5	81,4
23	49,4	77,6	48	51,3	80,8
24	57,7	91,0	49	61,5	91,0
25	56,4	87,8	50	61,5	91,3

8.2 Python scripts

Note these scripts are for viewing not coding. Check Github owned by j.meuwese@student.utwente.nl.

8.2.1 Readme

00_main_script (essential)
This script will take data from the Runelabs platform (cloud) and stores it in a CSV-file. The load in of the accelerometer and gyro meter are currently disabled but can be activated by uncommenting the lines of target script. It is important to give the information in which map the data has to be stored (this has to be

done before the script is running). If the user has access to 'Apple aparatuur Runelabs' the Watch_id, start time and endtime will be filled in automatically. The patient_id has to be given separately this is a safety measure. At the end the data will be plotted, this is to verify that there is data available in the period of interest.

01_merge_trem_sev_dys_all (essential)
This script takes the csv-files created using previous script, and combines these to one big file in excel format. This procedure is done automatically by calling a for loop that will run trough every patient and smartwatch it is important that all maps and files are in the same format as given for the first 25 patients. Otherwise it will be impossible to find the maps containing the datafiles.

02_avail_events_to_rest (non-essential)
If the events and availability are needer for the analyses this script can be helpful. But for the availability the regular datasets are automatically set to available thus calling for this information only shows at which the times the data was registered and when not. For the events this script can be useful to identify when the patient experiences tremor, dyskinesia or other symptoms. The disadvantage of this script is that due to the Pandas stopped supporting the 'append' function. Concat needs to be used instead. This only works when sys.exit is called in the function otherwise Python will create a dataframe of three dimensions.

03_merge_avail_events_to_rest (non-essential) Merges the events and availability with the merged dataset of script (01). This is done for each watch separately. Some patients have multiple watches.

04_UPDRS_Loadr.py (essential)
This script takes the UPDRS overview as input and stores all data in variables in Python. This script automatically registers of the patient is on or off during the test. The data of the patient from all visits can be used as input for other analyses. Or can be plotted at the end of this script.

05_merge_UPDRS_loadr_data_and_smartwatch_data (essential)
To compare the UPDRS and the smartwatch data around the given time of the test. Both need to be stored in the same file in the right format, This script combines these two different inputs and stores these in a new file called:'data_complete'. To train our algorithm these steps are crucial if you are only working with one of them (UPDRS or smartwatch data) this script will be less helpful. This script is written to work fully automatically.

06_pre_algorithm
This script takes the watch data as input and can be used to find the averages of the tremor and dyskinesia around the time of interest (when the UPDRS where taken). At the end it is possible to automatically save the averages in an Excel file but this can also be done manually. It only gives the tremor and dyskinesia as a result and not the patient or visit. Note: sometimes the columns change it is important that in df8 the first column equals time and the last column equals 'probability dyskinesia'.

07_x_algorithm (essential)
This script takes data_complete as input and is an algorithm that predicts the UPDRS score based on the smartwatch data. Currently not many datapoints are available, thus the accuracy of the algorithm is not that great but over time when more data points are available the accuracy will increase. The used

algorithm is a supervised machine learning function, called Randomforest (RF). It is based on a decision tree (DT) but takes the average of multiple DT's.

There are currently 3 scripts that have the 07 tag: 07_1, 07_2 and 07_3. These scripts are all algorithms developed by us, ordered from first consieved to last consieved. 07_1 Uses a single RF to calculate the UPDRS-scores. 07_2 uses a seperate RF per score (which amounts to 26 RF's) to calculate the MDS-UPDRS-scores. There is no significant difference between these two version, excet that 07_2 gives a narrower band of accuracies, but the mean accuracies are the same. 07_3 Uses hyperparameter tuning to first identify the best hyperparameters before training, but is otherwise nearly identical to 07_1. This increased the accuracy of the algorithm, but also increased the training times. We reccommend using 07_03 for analysing data in the future, as this algorithm will most likely give the best results for accuracy.

08_Analyse_complete_data_tabel (essential)
Gives a graphic interface of the results from a patient of a certain visit compared to another visit (user can choose visit of interest). And shows the difference between these two. The input is data_complete. And contains both UPDRS and smartwatch data.

09_graphics (essential)
This script takes the merged data from a patient of all watches and shows the mean tremor and dyskinesia for each hour. This can be helpful to get a better understanding of when the patient experiences certain symptoms.

8.2.2 00_main_script

```
Main_script
"""
TGO code voor het automatisch inladen van pt-data a.d.h.v. excel-sheet en cloud voor pandas 2.x
Door: Loet Schoenmakers, Job Meuwese & Stef Berendsen

Maakt gebruik van:
    Pandas v. 2.0.1.
    matplotlib v. 3.7.0
    runeq v. 0.11.0
    openpyxl 3.1.2

op basis van open-source code van rune-labs:
URL: https://github.com/rune-labs/opensource/tree/master/jupyter-notebook-template
"""

#%%
import io
import os
```

```
from datetime import datetime as dt
import pandas as pd
import matplotlib.pyplot as plt
import openpyxl
from runeq import Config, stream
from datetime import date

#defineer hieronder waar je de data wilt exporteren (BASE_PATH) & onder welke naam het
opgeslagen moet worden (PT_FILENAME)
BASE_PATH = 'E:\Datasets\ptxx_wtxx'
PT_FILENAME = 'ptxx_wtxx'

def make_full_df(accessor): #deze functie zorgt ervoor dat er een dataframe wordt aangemaakt
    """Loop through pages of API calls and append to a single dataframe"""
    df=pd.DataFrame()
    for page in accessor.iter_csv_text():
        page_df = pd.read_csv(io.StringIO(page))
        df = pd.DataFrame(page_df)
        df=pd.concat([df], ignore_index=True)
    return df

def get_watch_data(client, params, field, save_filepath=None):#deze functie zorgt eroor dat de
data eruit gehaald wordt
    """Makes API calls for watch data, saves to CSV and outputs dataframe"""
    fields = {
        #'accel': client.Accel(**params),
        #'rotation': client.Rotation(**params),
        #'heart rate': client.HeartRate(**params),
        'tremor': client.ProbabilitySymptom(
            symptom='tremor',
            **params),
        'tremor severity': client.ProbabilitySymptom(
            symptom='tremor',
            severity='*',
            **params),
        'dyskinesia': client.ProbabilitySymptom(
            symptom='dyskinesia',
            **params)
    }
```

```
    if field not in fields:
        raise ValueError('Please choose an existing field')

    accessor = fields[field]
    df = make_full_df(accessor)

    if save_filepath:
        df.to_csv(save_filepath, index=False)

    return df
#%% haal start- en eindtijden en de watch ID uit excel
pt_data = pd.read_excel('E:\Apple apparatuur in RuneLabs.xlsx') #laad excel bestand in

#noteer hieronder:
interesting_row = pt_data[pt_data['Study number'] == 'subxxx'] #de rij (pt) waaruit je het
wilt halen
number_of_watch = 'x' #van welke horloge je het wilt pakken

#verder haalt hij de watch ID, en start- en eindtijden uit het excelbestand. indien de
eindtijd nu is, pakt hij de tijd van je computer
watch_id_string = 'Watch ID' + number_of_watch
start_time = 'Start Time' + number_of_watch
end_time = 'End Time' + number_of_watch

watch_ID = interesting_row[watch_id_string]
watch_ID = watch_ID.to_string()
watch_ID = watch_ID[-8:]

start_time_date = interesting_row[start_time]
start_time_date = start_time_date.to_string()
start_time_date = start_time_date[-10:]
start_time_date = dt.strptime(start_time_date, '%Y-%m-%d') #%Y-M-D-H_Min

end_time_date = interesting_row[end_time]
end_time_date = end_time_date.to_string()
if 10< len(end_time_date) < 16:
    end_time_date = end_time_date[-10:]
    end_time_date = dt.strptime(end_time_date, '%Y-%m-%d')
```

```
else:

    end_time_date = end_time_date[-1:0]
    end_time_date = dt.today() #%Y-M-D-H_Min
    end_time_date = end_time_date.strftime('%Y-%m-%d')
    end_time_date = dt.strptime(end_time_date, '%Y-%m-%d') #%Y-M-D-H_Min

    end_time_date = end_time_date[-1:0]
    end_time_date = dt.today() #%Y-M-D-H_Min
    end_time_date = end_time_date.strftime('%Y-%m-%d')
    end_time_date = dt.strptime(end_time_date, '%Y-%m-%d') #%Y-M-D-H_Min
#%%
# set up client: haal de data van de cloud af en zet het in bestanden op de eerder aangewezen
plek

cfg = Config()
client = stream.V1Client(cfg)

params = {
    'patient_id': "",   #pt ID van runelabs
    'device_id': watch_ID,  # an Apple Watch
    # Use datetime to get timestamps from human-readable dates/times
    'start_time': start_time_date.timestamp(),
    'end_time': end_time_date.timestamp()
}

#we pakken alleen tremor probabilty, tremor severety en dyskinesia probability
#Retrieve data
# accel = get_watch_data(
#     client,
#     params,
#     'accel',
#     os.path.join(BASE_PATH, 'nb02_ex01_accel.csv')
# )
# rotation = get_watch_data(
#     client,
#     params,
#     'rotation',
#     os.path.join(BASE_PATH, 'nb02_ex01_rotation.csv')
```

```
# )
# heartrate = get_watch_data(
#     client,
#     params,
#     'heart rate',
#     os.path.join(BASE_PATH, 'nb02_ex01_heartrate.csv')
# )
tremor_prob = get_watch_data(
    client,
    params,
    'tremor',
    os.path.join(BASE_PATH, PT_FILENAME + "_tremor.csv")
)
tremor_severity = get_watch_data(
    client,
    params,
    'tremor severity',
    os.path.join(BASE_PATH, PT_FILENAME + "_tremor_severity.csv")
)
dyskinesia = get_watch_data(
    client,
    params,
    'dyskinesia',
    os.path.join(BASE_PATH, PT_FILENAME + "_dyskinesia.csv")
)
# def check_prob_data_availability(client, params):
#     """
#     Finds data availability of the (tremor and dyskinesia) probabilities within
#     some time frame
#     Outputs: dataframe with 0's and 1's for data availability per timestamp
#     """

#     accessor = client.ProbabilitySymptom(expression='availability(probability)',
symptom='tremor', **params)


#     for page in accessor.iter_json_availability():
#         df_page = pd.DataFrame(page['availability(probability)'])
#         df_page.insert(0, 'time', page['time'])
```

```
#          df=pd.DataFrame(df_page)
#          df = pd.concat([df], ignore_index=True)
#      return df
# availability = check_prob_data_availability(
#     client,
#     params,
#     )
# df = pd.DataFrame(availability)
# df.to_csv("E:\\Datasets\\pt09_wt03\\_availability.csv")
#%%
#Whole period
#xmin=start_time_date, # the one that doesn't change
#xmax=end_time_date, # the latest datetime in your dataset
#Or time of your choice
xmin=dt(2022, 11, 26, hour=12), # the one that doesn't change
xmax=dt(2022, 11, 26, hour=17) # the latest datetime in your dataset

#plotjes
fig, ax = plt.subplots(3, figsize=(14, 22))
#plot van termor probability, getal geeft aan hoeveel procent van de tijd de patient last had
van tremor
ax[0].plot(pd.to_datetime(tremor_prob.time, unit='s'), tremor_prob.probability,
drawstyle='steps-post')
ax[0].set_title('Tremor Probability')
ax[0].set_xlim([xmin,xmax])
#plot geeft aan hoe erg de tremor was tijdens de periode dat het aanwezig was
ax[1].stackplot(pd.to_datetime(tremor_severity.time, unit='s'),
                tremor_severity.none,
                tremor_severity.slight,
                tremor_severity.mild,
                tremor_severity.moderate,
                tremor_severity.strong,
                tremor_severity.unknown,
                labels=['none', 'slight', 'mild', 'moderate', 'strong', 'unknown'],
                colors=['#C8DEE0','#7FE0C4', '#DFE07F', '#ECB607', '#F42207', '#EFEBE7']
               )
ax[1].legend(bbox_to_anchor=(0.5, -.25), loc='lower center', ncol=6)
ax[1].set_title('Tremor Probability by Severity')
ax[1].set_xlim([xmin,xmax])
```

```
#plot geeft aan hoeveel procent van de tijd dyskinesia aanwezig was
ax[2].plot(pd.to_datetime(dyskinesia.time, unit='s'), dyskinesia.probability,
drawstyle='steps-post')
ax[2].set_title('Dyskinesia Probability')
ax[2].set_xlim([xmin,xmax])

fig.tight_layout(pad=2)
```

8.2.3 01_merge_trem_sev_dys_all

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 24 20:01:35 2023

@author: jobme & stefbe
"""

import os
import pandas as pd
#this script puts the smartwachtdata in columns according to each patient and watch_id
BASE_PATH = 'F:\Datasets'
PT_FILENAME = ['\ptxx', '\ptxx', '\ptxx', '\ptxx', '\ptxx', '\ptxx', '\ptxx', '\ptxx',
'\ptxx', '\ptxx', '\ptxx', '\ptxx', '\ptxx', '\ptxx', '\ptxx', '\ptxx', '\ptxx', '\ptxx',
'\ptxx', '\ptxx']
WATCH = ["_wtxx","_wtxx","_wtxx","_wtxx"]
# PT_FILENAME = '\ptxx_wtxx'
for a in range(len(PT_FILENAME)):
    for b in range(len(WATCH)):
        recieve= BASE_PATH+PT_FILENAME[a]+WATCH[b]+PT_FILENAME[a]+WATCH[b]
        recieve_lite = BASE_PATH+PT_FILENAME[a]+WATCH[b]
        store = recieve

        if os.path.exists(recieve_lite) == True:
            df1 = pd.read_csv(recieve+'_tremor.csv')
            df2 = pd.read_csv(recieve+'_tremor_severity.csv')
            df3 = pd.read_csv(recieve+'_dyskinesia.csv')

            df_merged = df1.merge(df2, on="time", how='outer')
            df_merged = df_merged.merge(df3, on="time", how='outer')
            df_merged = df_merged.rename(columns={'probability_x': 'probability_tremor',
```

```
'probability_y': 'probability_dyskinesia'})

            mdf = pd.DataFrame(df_merged) #merge done, data is now in a dataframe
            mdf = mdf[mdf["mild"].notnull()]#all columns who equal zero or dont exist are
deleted
            #
            avg_columns = [0,1,2,3,4,5,6,7]# new columns where average can be seen
            val = mdf.drop(columns="time")#from here the averages of each parameter is
calculated
            col = val.columns
            col = col.to_list()
            for x in range(len(col)):
                avg_columns[x] = sum(val[col[x]]) / len(val[col[x]])

            avg_columns.insert(0, "average")# adding the averages to the columns
            mdf.loc[len(mdf)] = avg_columns

            mdf.to_excel(store+'_merged.xlsx')
```

8.2.4 01b_merge_all_watches

```
    import os
import pandas as pd
#this script puts the smartwachtdata in columns according to each patient and watch_id
BASE_PATH = 'G:\Datasets'
PT_FILENAME = '\ptxx'
WATCH = ["_wtxx","_wtxx","_wtxx","_wtxx"]
watch_1="_wt01"
watch_2="_wt02"
watch_3="_wt03"
watch_4="_wt04"
merged="_merged.xlsx"
# PT_FILENAME = '\ptxx_wtxx'
store = 'G:\Datasets\overview'+PT_FILENAME+'_merged_all.xlsx'
number_of_watches = input("number_of_watches:")
if number_of_watches=='1':
    df1= pd.read_excel(BASE_PATH+PT_FILENAME+watch_1+PT_FILENAME+watch_1+merged)
    df_merged=df1
    print('one watch')
elif number_of_watches=='2':
```

```
    df1= pd.read_excel(BASE_PATH+PT_FILENAME+watch_1+PT_FILENAME+watch_1+merged)
    df2= pd.read_excel(BASE_PATH+PT_FILENAME+watch_2+PT_FILENAME+watch_2+merged)
    df_merged= pd.concat([df1,df2])
    print('two watches')
elif number_of_watches=='3':
      df1= pd.read_excel(BASE_PATH+PT_FILENAME+watch_1+PT_FILENAME+watch_1+merged)
      df2= pd.read_excel(BASE_PATH+PT_FILENAME+watch_2+PT_FILENAME+watch_2+merged)
      df3= pd.read_excel(BASE_PATH+PT_FILENAME+watch_3+PT_FILENAME+watch_3+merged)
      df_merged= pd.concat([df1,df2,df3])
      print ('three watches')
elif number_of_watches=='4':
      df1= pd.read_excel(BASE_PATH+PT_FILENAME+watch_1+PT_FILENAME+watch_1+merged)
      df2= pd.read_excel(BASE_PATH+PT_FILENAME+watch_2+PT_FILENAME+watch_2+merged)
      df3= pd.read_excel(BASE_PATH+PT_FILENAME+watch_3+PT_FILENAME+watch_3+merged)
      df4= pd.read_excel(BASE_PATH+PT_FILENAME+watch_4+PT_FILENAME+watch_4+merged)
      df_merged= pd.concat([df1,df2,df3,df4])#,ignore_index:=True)
      print('four watches')
else:
    print('kies zelf welke dataframes je wilt mergen, zet comments voor ongebruikte datframes')
    df1= pd.read_excel("")
    df2= pd.read_excel("")
    df3= pd.read_excel("")
    df4= pd.read_excel("")
    df_merged= pd.concat([df1,df2,df3,df4])

mdf = pd.DataFrame(df_merged) #merge done, data is now in a dataframe

#%%
mdf.to_excel(store)
```

8.2.5 02_avail_events_combi

```
    # -*- coding: utf-8 -*-
"""
Created on Fri May 26 09:53:45 2023

@author: loet-
"""
#for each cell: shift+return
import os
```

```
import datetime as dt
import pandas as pd
from runeq import Config, stream
import numpy as np
import sys
import io
import matplotlib.pyplot as plt

def get_events(client, params):
    """Makes API calls for events, outputs dataframe"""

    accessor = client.Event(**params)
    df=pd.DataFrame()
    for page in accessor.iter_json_data():
        df_page = pd.DataFrame(page['event'])
        global df_events

        df_events=df_page
        sys.exit()
        df = pd.concat([pd.DataFrame([df])], ignore_index=True)
    return df

cfg = Config()
client = stream.V1Client(cfg)
d=2
params = {
    'patient_id': 'patient_id',
    'device_id': 'watch_id',
    'start_time': dt.datetime(2022, 11, 23, hour=0, minute=30).timestamp(),
    'end_time': dt.datetime(2022, 11, 26, hour=2, minute=0).timestamp()
}

events = get_events(client, params)
#%%
def check_prob_data_availability(client, params):
    """
    Finds data availability of the (tremor and dyskinesia) probabilities within
    some time frame
    Outputs: dataframe with 0's and 1's for data availability per timestamp
```

```
    """

    accessor = client.ProbabilitySymptom(expression='availability(probability)',
symptom='tremor', **params)


    df = pd.DataFrame()
    for page in accessor.iter_json_availability():
        global df_page
        df_page = list(page['availability(probability)'])
        global df_avail
        df_avail = pd.DataFrame(df_page, columns=['probability'])
        df_avail.insert(0, 'time', page['time'])
        sys.exit()
        df = pd.concat(df_avail, ignore_index=True)
    return df

cfg = Config()
client = stream.V1Client(cfg)
params = {
    'patient_id': '',
    'device_id': '',  # an Apple Watch
    # Use datetime to get timestamps from human-readable dates/times
    'start_time': dt.datetime(2022, 11, 23, hour=0, minute=30).timestamp(),
    'end_time': dt.datetime(2022, 11, 25, hour=2, minute=0).timestamp()
}
BASE_PATH = 'G:\Datasets\pt09_wt03'
availability = check_prob_data_availability(
    client,
    params,
)
#%%export to excel
BASE_PATH = 'E:\Datasets'
PT_FILENAME = '\ptxx_wtxx'
recieve= BASE_PATH+PT_FILENAME+PT_FILENAME
store = BASE_PATH+PT_FILENAME+PT_FILENAME
df_events.to_excel(store+'events.xlsx')
df_avail.to_excel(store+'availability.xlsx')
```

8.2.6 03_merge_avail_events_to_rest

```
    # -*- coding: utf-8 -*-
"""
Created on Fri May 26 11:19:10 2023

@author: jobme
"""
import pandas as pd

BASE_PATH = 'G:\Datasets'
PT_FILENAME = '\ptxx_wtxx'
recieve= BASE_PATH+PT_FILENAME+PT_FILENAME
store = BASE_PATH+PT_FILENAME+PT_FILENAME
#%%
df4 = pd.read_excel(recieve+'merged.xlsx')
df5 = pd.read_excel(recieve+'events.xlsx')
df6 = pd.read_excel(recieve+'availability.xlsx')

df5['time']= df5['time'].astype('int')
df5['time']= (df5['time']//60 * 60)

df6['time']= df6['time'].astype('int')
df6['time']= (df6['time']//60 * 60)



df7_merged = df4.merge(df5, on="time", how='outer')
df7_merged = df7_merged.merge(df6, on="time", how='outer')
#df7_merged = df_merged.rename(columns={'probability_x': 'probability_tremor',
'probability_y': 'probability_dyskinesia'})
#%%
mmdf = pd.DataFrame(df7_merged)
mmdf.to_excel(store+'_merged_avail_events.xlsx')
```

8.2.7 04_UPDRS_Loadr

```
    # -*- coding: utf-8 -*-
"""
Created on Wed May 17 14:53:15 2023
```

```
@author: Job Meuwese, Loet Schoenmakers & Stef Berendsen

Purpose: read out the UPDRS-III overview.xlsx-file and place all data in a more structured
way, then export this data.

Uses the following libraries:
    pandas v: 2.0.1
    numpy v: 1.24.3
    matplotlib v: 3.7.1
"""

#import necessary scripts
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#%% upload and loading MDS-UPDRS overview

num_of_pt = xx #Define the amount of sheets in excell-1

pt_mds = pd.ExcelFile("F:/UPDRS-III overview.xlsx", engine='openpyxl') #load complete excel
file
Huge_data = {} #prepare dict, in which loose sheets of excell are placed
for x in range(num_of_pt): #per loose excell sheet:
    name="pt"+str(x) #create a name for dict
    Huge_data[name] = pd.read_excel(pt_mds, sheet_name=x) #read the corresponding excell table
and puts it in Huge_data dict
#if you want individual table of a patient, uncomment:
    #value = pd.read_excel(pt_mds, sheet_name=x)
    #globals()[f"pt{x}"] = value

#%% create UPDRS-dict
UPDRS = {} #define the UPDRS-dict
for x in range(num_of_pt): #per pt (en dus ook per excell sheet:)
    name="pt"+str(x) #define the  name
    temp = Huge_data[name] #temporary gets dataframe out the Huge_data Dict
    size_temp = temp.shape #temp gets format [length, width] from acquired dataframe

    if size_temp[0] == 0: #shows in UPDRS table that there is no data of this patient
```

```
        UPDRS[name, "notes"] = "this scorelist is empty"

    elif size_temp[0] >= 31: #visit 2

            if pd.isnull(temp.iloc[3, 5]) == True: #check whether the ON or OFF table has been
completed. this is done by looking at an empty field (rigidity head) in the ON
table, if it is empty it will read OFF, otherwise ON
                a = 4
            else:
                a = 0

            UPDRS[name, "v2_rig_head"] = temp.iloc[3,5-a]
            UPDRS[name, "v2_rig_right_arm"] = temp.iloc[4,5-a]
            UPDRS[name, "v2_rig_right_leg"] = temp.iloc[5,5-a]

            UPDRS[name, "v2_rig_left_arm"] = temp.iloc[4,7-a]
            UPDRS[name, "v2_rig_left_leg"] = temp.iloc[5,7-a]

            UPDRS[name, "v2_speech"] = temp.iloc[7,5-a]
            UPDRS[name, "v2_facial_expression"] = temp.iloc[8,5-a]

            UPDRS[name, "v2_right_finger_tap"] = temp.iloc[10,5-a]
            UPDRS[name, "v2_right_hand_movement"] = temp.iloc[11,5-a]
            UPDRS[name, "v2_right_pro_supination"] = temp.iloc[12,5-a]
            UPDRS[name, "v2_right_toe_tap"] = temp.iloc[13,5-a]
            UPDRS[name, "v2_right_leg_movement"] = temp.iloc[14,5-a]

            UPDRS[name, "v2_left_finger_tap"] = temp.iloc[10,7-a]
            UPDRS[name, "v2_left_hand_movement"] = temp.iloc[11,7-a]
            UPDRS[name, "v2_left_pro_supination"] = temp.iloc[12,7-a]
            UPDRS[name, "v2_left_toe_tap"] = temp.iloc[13,7-a]
            UPDRS[name, "v2_left_leg_movement"] = temp.iloc[14,7-a]

            UPDRS[name, "v2_raise_chair"] = temp.iloc[16,5-a]
            UPDRS[name, "v2_gait"] = temp.iloc[17,5-a]
            UPDRS[name, "v2_freezing"] = temp.iloc[18,5-a]
            UPDRS[name, "v2_postural_reflex"] = temp.iloc[19,5-a]
            UPDRS[name, "v2_posture"] = temp.iloc[20,5-a]
            UPDRS[name, "v2_spontaneous_move"] = temp.iloc[21,5-a]
```

```
            UPDRS[name, "v2_right_postural_tremor"] = temp.iloc[24,5-a]
            UPDRS[name, "v2_right_kinetic_tremor"] = temp.iloc[25,5-a]
            UPDRS[name, "v2_right_rest_tremor_amplitude_arm"] = temp.iloc[26,5-a]
            UPDRS[name, "v2_right_rest_tremor_amplitude_leg"] = temp.iloc[27,5-a]
            UPDRS[name, "v2_right_rest_tremor_constancy"] = temp.iloc[28,5-a]

            UPDRS[name, "v2_left_postural_tremor"] = temp.iloc[24,7-a]
            UPDRS[name, "v2_left_kinetic_tremor"] = temp.iloc[25,7-a]
            UPDRS[name, "v2_left_rest_tremor_amplitude_arm"] = temp.iloc[26,7-a]
            UPDRS[name, "v2_left_rest_tremor_amplitude_leg"] = temp.iloc[27,7-a]
            UPDRS[name, "v2_left_rest_tremor_constancy"] = temp.iloc[28,7-a]

            UPDRS[name, "v2_list_rig_tot"] =
(temp.iloc[3,5-a],temp.iloc[4,5-a],temp.iloc[5,5-a],temp.iloc[4,7-a],temp.iloc[5,7-
a])
            UPDRS[name, "v2_list_speech_facial"] = (temp.iloc[7,5-a],temp.iloc[8,5-a])
            UPDRS[name, "v2_list_brady_kin_right"] = (temp.iloc[10,5-a], temp.iloc[11,5-a],
temp.iloc[12,5-a], temp.iloc[13,5-a], temp.iloc[14,5-a])
            UPDRS[name, "v2_list_brady_kin_left"] = (temp.iloc[10,7-a], temp.iloc[11,7-a],
temp.iloc[12,7-a], temp.iloc[13,7-a], temp.iloc[14,7-a])
            UPDRS[name, "v2_list_brady_kin_tot"] = (UPDRS[name, "v2_list_brady_kin_left"] +
UPDRS[name, "v2_list_brady_kin_right"])
            UPDRS[name, "v2_list_gait_balance"] = (temp.iloc[16,5-a], temp.iloc[17,5-a],
temp.iloc[18,5-a], temp.iloc[19,5-a], temp.iloc[20,5-a], temp.iloc[21,5-a])
            UPDRS[name, "v2_list_tremor_right"]= (temp.iloc[24,5-a], temp.iloc[25,5-a],
temp.iloc[26,5-a], temp.iloc[27,5-a], temp.iloc[28,5-a])
            UPDRS[name, "v2_list_tremor_left"]= (temp.iloc[24,7-a], temp.iloc[25,7-a],
temp.iloc[26,7-a], temp.iloc[27,7-a], temp.iloc[28,7-a] )
            UPDRS[name, "v2_list_tremor_tot"]= (UPDRS[name, "v2_list_tremor_right"] +
UPDRS[name, "v2_list_tremor_left"])
            UPDRS[name, "v2_list_globaal"] = (UPDRS[name, "v2_list_rig_tot"] + UPDRS[name,
"v2_list_speech_facial"] + UPDRS[name, "v2_list_brady_kin_tot"] + UPDRS[name,
"v2_list_gait_balance"] + UPDRS[name, "v2_list_tremor_tot"])

            if size_temp[0] >= 64: #visit 3
                if pd.isnull(temp.iloc[36, 5]) == True: #check whether the ON or OFF table has
been completed. this is done by looking at an empty field (rigidity head) in
the ON table, if it is empty it will read OFF, otherwise ON
```

```
    a = 4
else:
    a = 0

UPDRS[name, "v3_rig_head"] = temp.iloc[36,5-a]
UPDRS[name, "v3_rig_right_arm"] = temp.iloc[37,5-a]
UPDRS[name, "v3_rig_right_leg"] = temp.iloc[38,5-a]
UPDRS[name, "v3_rig_left_arm"] = temp.iloc[37,7-a]
UPDRS[name, "v3_rig_left_leg"] = temp.iloc[38,7-a]

UPDRS[name, "v3_speech"] = temp.iloc[40,5-a]
UPDRS[name, "v3_facial_expression"] = temp.iloc[41,5-a]

UPDRS[name, "v3_right_finger_tap"] = temp.iloc[43,5-a]
UPDRS[name, "v3_right_hand_movement"] = temp.iloc[44,5-a]
UPDRS[name, "v3_right_pro_supination"] = temp.iloc[45,5-a]
UPDRS[name, "v3_right_toe_tap"] = temp.iloc[46,5-a]
UPDRS[name, "v3_right_leg_movement"] = temp.iloc[47,5-a]

UPDRS[name, "v3_left_finger_tap"] = temp.iloc[43,7-a]
UPDRS[name, "v3_left_hand_movement"] = temp.iloc[44,7-a]
UPDRS[name, "v3_left_pro_supination"] = temp.iloc[45,7-a]
UPDRS[name, "v3_left_toe_tap"] = temp.iloc[46,7-a]
UPDRS[name, "v3_left_leg_movement"] = temp.iloc[47,7-a]

UPDRS[name, "v3_raise_chair"] = temp.iloc[49,5-a]
UPDRS[name, "v3_gait"] = temp.iloc[50,5-a]
UPDRS[name, "v3_freezing"] = temp.iloc[51,5-a]
UPDRS[name, "v3_postural_reflex"] = temp.iloc[52,5-a]
UPDRS[name, "v3_posture"] = temp.iloc[53,5-a]
UPDRS[name, "v3_spontaneous_move"] = temp.iloc[54,5-a]

UPDRS[name, "v3_right_postural_tremor"] = temp.iloc[57,5-a]
UPDRS[name, "v3_right_kinetic_tremor"] = temp.iloc[58,5-a]
UPDRS[name, "v3_right_rest_tremor_amplitude_arm"] = temp.iloc[59,5-a]
UPDRS[name, "v3_right_rest_tremor_amplitude_leg"] = temp.iloc[60,5-a]
UPDRS[name, "v3_right_rest_tremor_constancy"] = temp.iloc[61,5-a]

UPDRS[name, "v3_left_postural_tremor"] = temp.iloc[57,7-a]
```

```
                UPDRS[name, "v3_left_kinetic_tremor"] = temp.iloc[58,7-a]
                UPDRS[name, "v3_left_rest_tremor_amplitude_arm"] = temp.iloc[59,7-a]
                UPDRS[name, "v3_left_rest_tremor_amplitude_leg"] = temp.iloc[60,7-a]
                UPDRS[name, "v3_left_rest_tremor_constancy"] = temp.iloc[61,7-a]

                UPDRS[name, "v3_list_rig_tot"] =
(temp.iloc[36,5-a],temp.iloc[37,5-a],temp.iloc[38,5-a],temp.iloc[37,7-a],temp.i
loc[38,7-a])
                UPDRS[name, "v3_list_speech_facial"] = (temp.iloc[40,5-a],temp.iloc[41,5-a])
                UPDRS[name, "v3_list_brady_kin_right"] = (temp.iloc[43,5-a],
temp.iloc[44,5-a], temp.iloc[45,5-a], temp.iloc[46,5-a], temp.iloc[47,5-a])
                UPDRS[name, "v3_list_brady_kin_left"] = (temp.iloc[43,7-a], temp.iloc[44,7-a],
temp.iloc[45,7-a], temp.iloc[46,7-a], temp.iloc[47,7-a])
                UPDRS[name, "v3_list_brady_kin_tot"] = (UPDRS[name, "v3_list_brady_kin_left"]
+ UPDRS[name, "v3_list_brady_kin_right"])
                UPDRS[name, "v3_list_gait_balance"] = (temp.iloc[49,5-a], temp.iloc[50,5-a],
temp.iloc[51,5-a], temp.iloc[52,5-a], temp.iloc[53,5-a], temp.iloc[54,5-a])
                UPDRS[name, "v3_list_tremor_right"]= (temp.iloc[57,5-a], temp.iloc[58,5-a],
temp.iloc[59,5-a], temp.iloc[60,5-a], temp.iloc[61,5-a])
                UPDRS[name, "v3_list_tremor_left"]= (temp.iloc[57,7-a], temp.iloc[58,7-a],
temp.iloc[59,7-a], temp.iloc[60,7-a], temp.iloc[61,7-a] )
                UPDRS[name, "v3_list_tremor_tot"]= (UPDRS[name, "v3_list_tremor_right"] +
UPDRS[name, "v3_list_tremor_left"])
                UPDRS[name, "v3_list_globaal"] = (UPDRS[name, "v3_list_rig_tot"] + UPDRS[name,
"v3_list_speech_facial"] + UPDRS[name, "v3_list_brady_kin_tot"] + UPDRS[name,
"v3_list_gait_balance"] + UPDRS[name, "v3_list_tremor_tot"])

                if size_temp[0] >= 97: #visit 4
                    if pd.isnull(temp.iloc[69, 5]) == True: #check whether the ON or OFF table
has been completed. this is done by looking at an empty field (rigidity
head) in the ON table, if it is empty it will read OFF, otherwise ON
                        a = 4
                    else:
                        a = 0

                    UPDRS[name, "v4_rig_head"] = temp.iloc[69,5-a]
                    UPDRS[name, "v4_rig_right_arm"] = temp.iloc[70,5-a]
                    UPDRS[name, "v4_rig_right_leg"] = temp.iloc[71,5-a]
                    UPDRS[name, "v4_rig_left_arm"] = temp.iloc[70,7-a]
```

```
                    UPDRS[name, "v4_rig_left_leg"] = temp.iloc[71,7-a]

                    UPDRS[name, "v4_speech"] = temp.iloc[73,5-a]
                    UPDRS[name, "v4_facial_expression"] = temp.iloc[74,5-a]

                    UPDRS[name, "v4_right_finger_tap"] = temp.iloc[76,5-a]
                    UPDRS[name, "v4_right_hand_movement"] = temp.iloc[77,5-a]
                    UPDRS[name, "v4_right_pro_supination"] = temp.iloc[78,5-a]
                    UPDRS[name, "v4_right_toe_tap"] = temp.iloc[79,5-a]
                    UPDRS[name, "v4_right_leg_movement"] = temp.iloc[80,5-a]

                    UPDRS[name, "v4_left_finger_tap"] = temp.iloc[76,7-a]
                    UPDRS[name, "v4_left_hand_movement"] = temp.iloc[77,7-a]
                    UPDRS[name, "v4_left_pro_supination"] = temp.iloc[78,7-a]
                    UPDRS[name, "v4_left_toe_tap"] = temp.iloc[79,7-a]
                    UPDRS[name, "v4_left_leg_movement"] = temp.iloc[80,7-a]

                    UPDRS[name, "v4_raise_chair"] = temp.iloc[82,5-a]
                    UPDRS[name, "v4_gait"] = temp.iloc[83,5-a]
                    UPDRS[name, "v4_freezing"] = temp.iloc[84,5-a]
                    UPDRS[name, "v4_postural_reflex"] = temp.iloc[85,5-a]
                    UPDRS[name, "v4_posture"] = temp.iloc[86,5-a]
                    UPDRS[name, "v4_spontaneous_move"] = temp.iloc[87,5-a]

                    UPDRS[name, "v4_right_postural_tremor"] = temp.iloc[90,5-a]
                    UPDRS[name, "v4_right_kinetic_tremor"] = temp.iloc[91,5-a]
                    UPDRS[name, "v4_right_rest_tremor_amplitude_arm"] = temp.iloc[92,5-a]
                    UPDRS[name, "v4_right_rest_tremor_amplitude_leg"] = temp.iloc[93,5-a]
                    UPDRS[name, "v4_right_rest_tremor_constancy"] = temp.iloc[94,5-a]

                    UPDRS[name, "v4_left_postural_tremor"] = temp.iloc[90,7-a]
                    UPDRS[name, "v4_left_kinetic_tremor"] = temp.iloc[91,7-a]
                    UPDRS[name, "v4_left_rest_tremor_amplitude_arm"] = temp.iloc[92,7-a]
                    UPDRS[name, "v4_left_rest_tremor_amplitude_leg"] = temp.iloc[93,7-a]
                    UPDRS[name, "v4_left_rest_tremor_constancy"] = temp.iloc[94,7-a]

                    UPDRS[name, "v4_list_rig_tot"] =
(temp.iloc[69,5-a],temp.iloc[70,5-a],temp.iloc[71,5-a],temp.iloc[70,7-a],te
mp.iloc[71,7-a])
```

```
                    UPDRS[name, "v4_list_speech_facial"] =
(temp.iloc[73,5-a],temp.iloc[74,5-a])
                    UPDRS[name, "v4_list_brady_kin_right"] = (temp.iloc[76,5-a],
temp.iloc[77,5-a], temp.iloc[78,5-a], temp.iloc[79,5-a], temp.iloc[80,5-a])
                    UPDRS[name, "v4_list_brady_kin_left"] = (temp.iloc[76,7-a],
temp.iloc[77,7-a], temp.iloc[78,7-a], temp.iloc[79,7-a], temp.iloc[80,7-a])
                    UPDRS[name, "v4_list_brady_kin_tot"] = (UPDRS[name,
"v4_list_brady_kin_left"] + UPDRS[name, "v4_list_brady_kin_right"])
                    UPDRS[name, "v4_list_gait_balance"] = (temp.iloc[82,5-a],
temp.iloc[83,5-a], temp.iloc[84,5-a], temp.iloc[85,5-a],
temp.iloc[86,5-a], temp.iloc[87,5-a])
                    UPDRS[name, "v4_list_tremor_right"]= (temp.iloc[90,5-a],
temp.iloc[91,5-a], temp.iloc[92,5-a], temp.iloc[93,5-a], temp.iloc[94,5-a])
                    UPDRS[name, "v4_list_tremor_left"]= (temp.iloc[90,7-a], temp.iloc[91,7-a],
temp.iloc[92,7-a], temp.iloc[93,7-a], temp.iloc[94,7-a] )
                    UPDRS[name, "v4_list_tremor_tot"]= (UPDRS[name, "v4_list_tremor_right"] +
UPDRS[name, "v4_list_tremor_left"])
                    UPDRS[name, "v4_list_globaal"] = (UPDRS[name, "v4_list_rig_tot"] +
UPDRS[name, "v4_list_speech_facial"] + UPDRS[name,
"v4_list_brady_kin_tot"] + UPDRS[name, "v4_list_gait_balance"] +
UPDRS[name, "v4_list_tremor_tot"])

                    if size_temp[0] >= 229: #haal alle data op van de 5de visit, met al zijn
meetmomenten
                        # visit 5_1
                        if pd.isnull(temp.iloc[102, 5]) == True: #check whether the ON or OFF
table has been completed. this is done by looking at an empty field
(rigidity head) in the ON table, if it is empty it will read OFF,
otherwise ON
                            a = 4
                        else:
                            a = 0

                        UPDRS[name, "v5_1_rig_head"] = temp.iloc[102,5-a]
                        UPDRS[name, "v5_1_rig_right_arm"] = temp.iloc[103,5-a]
                        UPDRS[name, "v5_1_rig_right_leg"] = temp.iloc[104,5-a]
                        UPDRS[name, "v5_1_rig_left_arm"] = temp.iloc[103,7-a]
                        UPDRS[name, "v5_1_rig_left_leg"] = temp.iloc[104,7-a]
```

```
                        UPDRS[name, "v5_1_speech"] = temp.iloc[106,5-a]
                        UPDRS[name, "v5_1_facial_expression"] = temp.iloc[107,5-a]

                        UPDRS[name, "v5_1_right_finger_tap"] = temp.iloc[109,5-a]
                        UPDRS[name, "v5_1_right_hand_movement"] = temp.iloc[110,5-a]
                        UPDRS[name, "v5_1_right_pro_supination"] = temp.iloc[111,5-a]
                        UPDRS[name, "v5_1_right_toe_tap"] = temp.iloc[112,5-a]
                        UPDRS[name, "v5_1_right_leg_movement"] = temp.iloc[113,5-a]

                        UPDRS[name, "v5_1_left_finger_tap"] = temp.iloc[109,7-a]
                        UPDRS[name, "v5_1_left_hand_movement"] = temp.iloc[110,7-a]
                        UPDRS[name, "v5_1_left_pro_supination"] = temp.iloc[111,7-a]
                        UPDRS[name, "v5_1_left_toe_tap"] = temp.iloc[112,7-a]
                        UPDRS[name, "v5_1_left_leg_movement"] = temp.iloc[113,7-a]

                        UPDRS[name, "v5_1_raise_chair"] = temp.iloc[115,5-a]
                        UPDRS[name, "v5_1_gait"] = temp.iloc[116,5-a]
                        UPDRS[name, "v5_1_freezing"] = temp.iloc[117,5-a]
                        UPDRS[name, "v5_1_postural_reflex"] = temp.iloc[118,5-a]
                        UPDRS[name, "v5_1_posture"] = temp.iloc[119,5-a]
                        UPDRS[name, "v5_1_spontaneous_move"] = temp.iloc[120,5-a]

                        UPDRS[name, "v5_1_right_postural_tremor"] = temp.iloc[123,5-a]
                        UPDRS[name, "v5_1_right_kinetic_tremor"] = temp.iloc[124,5-a]
                        UPDRS[name, "v5_1_right_rest_tremor_amplitude_arm"] =
temp.iloc[125,5-a]
                        UPDRS[name, "v5_1_right_rest_tremor_amplitude_leg"] =
temp.iloc[126,5-a]
                        UPDRS[name, "v5_1_right_rest_tremor_constancy"] = temp.iloc[127,5-a]

                        UPDRS[name, "v5_1_left_postural_tremor"] = temp.iloc[123,7-a]
                        UPDRS[name, "v5_1_left_kinetic_tremor"] = temp.iloc[124,7-a]
                        UPDRS[name, "v5_1_left_rest_tremor_amplitude_arm"] = temp.iloc[125,7-a]
                        UPDRS[name, "v5_1_left_rest_tremor_amplitude_leg"] = temp.iloc[126,7-a]
                        UPDRS[name, "v5_1_left_rest_tremor_constancy"] = temp.iloc[127,7-a]

                        UPDRS[name, "v5_1_list_rig_tot"] =
(temp.iloc[102,5-a],temp.iloc[103,5-a],temp.iloc[104,5-a],temp.iloc[103
,7-a],temp.iloc[104,7-a])
```

```
                        UPDRS[name, "v5_1_list_speech_facial"] =
(temp.iloc[106,5-a],temp.iloc[107,5-a])
                        UPDRS[name, "v5_1_list_brady_kin_right"] = (temp.iloc[109,5-a],
temp.iloc[110,5-a], temp.iloc[111,5-a], temp.iloc[112,5-a],
temp.iloc[113,5-a])
                        UPDRS[name, "v5_1_list_brady_kin_left"] = (temp.iloc[109,7-a],
temp.iloc[110,7-a], temp.iloc[111,7-a], temp.iloc[112,7-a],
temp.iloc[113,7-a])
                        UPDRS[name, "v5_1_list_brady_kin_tot"] = (UPDRS[name,
"v5_1_list_brady_kin_left"] + UPDRS[name, "v5_1_list_brady_kin_right"])
                        UPDRS[name, "v5_1_list_gait_balance"] = (temp.iloc[115,5-a],
temp.iloc[116,5-a], temp.iloc[117,5-a], temp.iloc[118,5-a],
temp.iloc[119,5-a], temp.iloc[120,5-a])
                        UPDRS[name, "v5_1_list_tremor_right"]= (temp.iloc[123,5-a],
temp.iloc[124,5-a], temp.iloc[125,5-a], temp.iloc[126,5-a],
temp.iloc[127,5-a])
                        UPDRS[name, "v5_1_list_tremor_left"]= (temp.iloc[123,7-a],
temp.iloc[124,7-a], temp.iloc[125,7-a], temp.iloc[126,7-a],
temp.iloc[127,7-a] )
                        UPDRS[name, "v5_1_list_tremor_tot"]= (UPDRS[name,
"v5_1_list_tremor_right"] + UPDRS[name, "v5_1_list_tremor_left"])
                        UPDRS[name, "v5_1_list_globaal"] = (UPDRS[name, "v5_1_list_rig_tot"] +
UPDRS[name, "v5_1_list_speech_facial"] + UPDRS[name,
"v5_1_list_brady_kin_tot"] + UPDRS[name, "v5_1_list_gait_balance"] +
UPDRS[name, "v5_1_list_tremor_tot"])



                        # visit 5_2
                        if pd.isnull(temp.iloc[135, 5]) == True:  #check whether the ON or OFF
table has been completed. this is done by looking at an empty field
(rigidity head) in the ON table, if it is empty it will read OFF,
otherwise ON
                            a = 4
                        else:
                            a = 0

                        UPDRS[name, "v5_2_rig_head"] = temp.iloc[135,5-a]
                        UPDRS[name, "v5_2_rig_right_arm"] = temp.iloc[136,5-a]
                        UPDRS[name, "v5_2_rig_right_leg"] = temp.iloc[137,5-a]
```

```
                        UPDRS[name, "v5_2_rig_left_arm"] = temp.iloc[136,7-a]
                        UPDRS[name, "v5_2_rig_left_leg"] = temp.iloc[137,7-a]

                        UPDRS[name, "v5_2_speech"] = temp.iloc[139,5-a]
                        UPDRS[name, "v5_2_facial_expression"] = temp.iloc[140,5-a]

                        UPDRS[name, "v5_2_right_finger_tap"] = temp.iloc[142,5-a]
                        UPDRS[name, "v5_2_right_hand_movement"] = temp.iloc[143,5-a]
                        UPDRS[name, "v5_2_right_pro_supination"] = temp.iloc[144,5-a]
                        UPDRS[name, "v5_2_right_toe_tap"] = temp.iloc[145,5-a]
                        UPDRS[name, "v5_2_right_leg_movement"] = temp.iloc[146,5-a]

                        UPDRS[name, "v5_2_left_finger_tap"] = temp.iloc[142,7-a]
                        UPDRS[name, "v5_2_left_hand_movement"] = temp.iloc[143,7-a]
                        UPDRS[name, "v5_2_left_pro_supination"] = temp.iloc[144,7-a]
                        UPDRS[name, "v5_2_left_toe_tap"] = temp.iloc[145,7-a]
                        UPDRS[name, "v5_2_left_leg_movement"] = temp.iloc[146,7-a]

                        UPDRS[name, "v5_2_raise_chair"] = temp.iloc[148,5-a]
                        UPDRS[name, "v5_2_gait"] = temp.iloc[149,5-a]
                        UPDRS[name, "v5_2_freezing"] = temp.iloc[150,5-a]
                        UPDRS[name, "v5_2_postural_reflex"] = temp.iloc[151,5-a]
                        UPDRS[name, "v5_2_posture"] = temp.iloc[152,5-a]
                        UPDRS[name, "v5_2_spontaneous_move"] = temp.iloc[153,5-a]

                        UPDRS[name, "v5_2_right_postural_tremor"] = temp.iloc[156,5-a]
                        UPDRS[name, "v5_2_right_kinetic_tremor"] = temp.iloc[157,5-a]
                        UPDRS[name, "v5_2_right_rest_tremor_amplitude_arm"] =
temp.iloc[158,5-a]
                        UPDRS[name, "v5_2_right_rest_tremor_amplitude_leg"] =
temp.iloc[159,5-a]
                        UPDRS[name, "v5_2_right_rest_tremor_constancy"] = temp.iloc[160,5-a]

                        UPDRS[name, "v5_2_left_postural_tremor"] = temp.iloc[156,7-a]
                        UPDRS[name, "v5_2_left_kinetic_tremor"] = temp.iloc[157,7-a]
                        UPDRS[name, "v5_2_left_rest_tremor_amplitude_arm"] = temp.iloc[158,7-a]
                        UPDRS[name, "v5_2_left_rest_tremor_amplitude_leg"] = temp.iloc[159,7-a]
                        UPDRS[name, "v5_2_left_rest_tremor_constancy"] = temp.iloc[160,7-a]
```

```
                        UPDRS[name, "v5_2_list_rig_tot"] =
(temp.iloc[135,5-a],temp.iloc[136,5-a],temp.iloc[137,5-a],temp.iloc[136
,7-a],temp.iloc[137,7-a])
                        UPDRS[name, "v5_2_list_speech_facial"] =
(temp.iloc[139,5-a],temp.iloc[140,5-a])
                        UPDRS[name, "v5_2_list_brady_kin_right"] = (temp.iloc[142,5-a],
temp.iloc[143,5-a], temp.iloc[144,5-a], temp.iloc[145,5-a],
temp.iloc[146,5-a])
                        UPDRS[name, "v5_2_list_brady_kin_left"] = (temp.iloc[142,7-a],
temp.iloc[143,7-a], temp.iloc[144,7-a], temp.iloc[145,7-a],
temp.iloc[146,7-a])
                        UPDRS[name, "v5_2_list_brady_kin_tot"] = (UPDRS[name,
"v5_2_list_brady_kin_left"] + UPDRS[name, "v5_2_list_brady_kin_right"])
                        UPDRS[name, "v5_2_list_gait_balance"] = (temp.iloc[148,5-a],
temp.iloc[149,5-a], temp.iloc[150,5-a], temp.iloc[151,5-a],
temp.iloc[152,5-a], temp.iloc[153,5-a])
                        UPDRS[name, "v5_2_list_tremor_right"]= (temp.iloc[156,5-a],
temp.iloc[157,5-a], temp.iloc[158,5-a], temp.iloc[159,5-a],
temp.iloc[160,5-a])
                        UPDRS[name, "v5_2_list_tremor_left"]= (temp.iloc[156,7-a],
temp.iloc[157,7-a], temp.iloc[158,7-a], temp.iloc[159,7-a],
temp.iloc[160,7-a] )
                        UPDRS[name, "v5_2_list_tremor_tot"]= (UPDRS[name,
"v5_2_list_tremor_right"] + UPDRS[name, "v5_2_list_tremor_left"])
                        UPDRS[name, "v5_2_list_globaal"] = (UPDRS[name, "v5_2_list_rig_tot"] +
UPDRS[name, "v5_2_list_speech_facial"] + UPDRS[name,
"v5_2_list_brady_kin_tot"] + UPDRS[name, "v5_2_list_gait_balance"] +
UPDRS[name, "v5_2_list_tremor_tot"])

                        # visit 5_3
                        if pd.isnull(temp.iloc[168, 5]) == True: #check whether the ON or OFF
table has been completed. this is done by looking at an empty field
(rigidity head) in the ON table, if it is empty it will read OFF,
otherwise ON
                            a = 4
                        else:
                            a = 0

                        UPDRS[name, "v5_3_rig_head"] = temp.iloc[168,5-a]
```

```
                        UPDRS[name, "v5_3_rig_right_arm"] = temp.iloc[169,5-a]
                        UPDRS[name, "v5_3_rig_right_leg"] = temp.iloc[170,5-a]
                        UPDRS[name, "v5_3_rig_left_arm"] = temp.iloc[169,7-a]
                        UPDRS[name, "v5_3_rig_left_leg"] = temp.iloc[170,7-a]

                        UPDRS[name, "v5_3_speech"] = temp.iloc[172,5-a]
                        UPDRS[name, "v5_3_facial_expression"] = temp.iloc[173,5-a]

                        UPDRS[name, "v5_3_right_finger_tap"] = temp.iloc[175,5-a]
                        UPDRS[name, "v5_3_right_hand_movement"] = temp.iloc[176,5-a]
                        UPDRS[name, "v5_3_right_pro_supination"] = temp.iloc[177,5-a]
                        UPDRS[name, "v5_3_right_toe_tap"] = temp.iloc[178,5-a]
                        UPDRS[name, "v5_3_right_leg_movement"] = temp.iloc[179,5-a]

                        UPDRS[name, "v5_3_left_finger_tap"] = temp.iloc[175,7-a]
                        UPDRS[name, "v5_3_left_hand_movement"] = temp.iloc[176,7-a]
                        UPDRS[name, "v5_3_left_pro_supination"] = temp.iloc[177,7-a]
                        UPDRS[name, "v5_3_left_toe_tap"] = temp.iloc[178,7-a]
                        UPDRS[name, "v5_3_left_leg_movement"] = temp.iloc[179,7-a]

                        UPDRS[name, "v5_3_raise_chair"] = temp.iloc[181,5-a]
                        UPDRS[name, "v5_3_gait"] = temp.iloc[182,5-a]
                        UPDRS[name, "v5_3_freezing"] = temp.iloc[183,5-a]
                        UPDRS[name, "v5_3_postural_reflex"] = temp.iloc[184,5-a]
                        UPDRS[name, "v5_3_posture"] = temp.iloc[185,5-a]
                        UPDRS[name, "v5_3_spontaneous_move"] = temp.iloc[186,5-a]

                        UPDRS[name, "v5_3_right_postural_tremor"] = temp.iloc[189,5-a]
                        UPDRS[name, "v5_3_right_kinetic_tremor"] = temp.iloc[190,5-a]
                        UPDRS[name, "v5_3_right_rest_tremor_amplitude_arm"] =
temp.iloc[191,5-a]
                        UPDRS[name, "v5_3_right_rest_tremor_amplitude_leg"] =
temp.iloc[192,5-a]
                        UPDRS[name, "v5_3_right_rest_tremor_constancy"] = temp.iloc[193,5-a]

                        UPDRS[name, "v5_3_left_postural_tremor"] = temp.iloc[189,7-a]
                        UPDRS[name, "v5_3_left_kinetic_tremor"] = temp.iloc[190,7-a]
                        UPDRS[name, "v5_3_left_rest_tremor_amplitude_arm"] = temp.iloc[191,7-a]
                        UPDRS[name, "v5_3_left_rest_tremor_amplitude_leg"] = temp.iloc[192,7-a]
```

```
                        UPDRS[name, "v5_3_left_rest_tremor_constancy"] = temp.iloc[193,7-a]

                        UPDRS[name, "v5_3_list_rig_tot"] =
(temp.iloc[168,5-a],temp.iloc[169,5-a],temp.iloc[170,5-a],temp.iloc[169
,7-a],temp.iloc[170,7-a])
                        UPDRS[name, "v5_3_list_speech_facial"] =
(temp.iloc[172,5-a],temp.iloc[173,5-a])
                        UPDRS[name, "v5_3_list_brady_kin_right"] = (temp.iloc[175,5-a],
temp.iloc[176,5-a], temp.iloc[177,5-a], temp.iloc[178,5-a],
temp.iloc[179,5-a])
                        UPDRS[name, "v5_3_list_brady_kin_left"] = (temp.iloc[175,7-a],
temp.iloc[176,7-a], temp.iloc[177,7-a], temp.iloc[178,7-a],
temp.iloc[179,7-a])
                        UPDRS[name, "v5_3_list_brady_kin_tot"] = (UPDRS[name,
"v5_3_list_brady_kin_left"] + UPDRS[name, "v5_3_list_brady_kin_right"])
                        UPDRS[name, "v5_3_list_gait_balance"] = (temp.iloc[181,5-a],
temp.iloc[182,5-a], temp.iloc[183,5-a], temp.iloc[184,5-a],
temp.iloc[185,5-a], temp.iloc[186,5-a])
                        UPDRS[name, "v5_3_list_tremor_right"]= (temp.iloc[189,5-a],
temp.iloc[190,5-a], temp.iloc[191,5-a], temp.iloc[192,5-a],
temp.iloc[193,5-a])
                        UPDRS[name, "v5_3_list_tremor_left"]= (temp.iloc[189,7-a],
temp.iloc[190,7-a], temp.iloc[191,7-a], temp.iloc[192,7-a],
temp.iloc[193,7-a] )
                        UPDRS[name, "v5_3_list_tremor_tot"]= (UPDRS[name,
"v5_3_list_tremor_right"] + UPDRS[name, "v5_3_list_tremor_left"])
                        UPDRS[name, "v5_3_list_globaal"] = (UPDRS[name, "v5_3_list_rig_tot"] +
UPDRS[name, "v5_3_list_speech_facial"] + UPDRS[name,
"v5_3_list_brady_kin_tot"] + UPDRS[name, "v5_3_list_gait_balance"] +
UPDRS[name, "v5_3_list_tremor_tot"])

                        # visit 5_4
                        if pd.isnull(temp.iloc[201, 5]) == True: #check whether the ON or OFF
table has been completed. this is done by looking at an empty field
(rigidity head) in the ON table, if it is empty it will read OFF,
otherwise ON
                            a = 4
                        else:
                            a = 0
```

```
                        UPDRS[name, "v5_4_rig_head"] = temp.iloc[201,5-a]
                        UPDRS[name, "v5_4_rig_right_arm"] = temp.iloc[202,5-a]
                        UPDRS[name, "v5_4_rig_right_leg"] = temp.iloc[203,5-a]
                        UPDRS[name, "v5_4_rig_left_arm"] = temp.iloc[202,7-a]
                        UPDRS[name, "v5_4_rig_left_leg"] = temp.iloc[203,7-a]

                        UPDRS[name, "v5_4_speech"] = temp.iloc[205,5-a]
                        UPDRS[name, "v5_4_facial_expression"] = temp.iloc[206,5-a]

                        UPDRS[name, "v5_4_right_finger_tap"] = temp.iloc[208,5-a]
                        UPDRS[name, "v5_4_right_hand_movement"] = temp.iloc[209,5-a]
                        UPDRS[name, "v5_4_right_pro_supination"] = temp.iloc[210,5-a]
                        UPDRS[name, "v5_4_right_toe_tap"] = temp.iloc[211,5-a]
                        UPDRS[name, "v5_4_right_leg_movement"] = temp.iloc[212,5-a]

                        UPDRS[name, "v5_4_left_finger_tap"] = temp.iloc[208,7-a]
                        UPDRS[name, "v5_4_left_hand_movement"] = temp.iloc[209,7-a]
                        UPDRS[name, "v5_4_left_pro_supination"] = temp.iloc[210,7-a]
                        UPDRS[name, "v5_4_left_toe_tap"] = temp.iloc[211,7-a]
                        UPDRS[name, "v5_4_left_leg_movement"] = temp.iloc[212,7-a]

                        UPDRS[name, "v5_4_raise_chair"] = temp.iloc[214,5-a]
                        UPDRS[name, "v5_4_gait"] = temp.iloc[215,5-a]
                        UPDRS[name, "v5_4_freezing"] = temp.iloc[216,5-a]
                        UPDRS[name, "v5_4_postural_reflex"] = temp.iloc[217,5-a]
                        UPDRS[name, "v5_4_posture"] = temp.iloc[218,5-a]
                        UPDRS[name, "v5_4_spontaneous_move"] = temp.iloc[219,5-a]

                        UPDRS[name, "v5_4_right_postural_tremor"] = temp.iloc[222,5-a]
                        UPDRS[name, "v5_4_right_kinetic_tremor"] = temp.iloc[223,5-a]
                        UPDRS[name, "v5_4_right_rest_tremor_amplitude_arm"] =
temp.iloc[224,5-a]
                        UPDRS[name, "v5_4_right_rest_tremor_amplitude_leg"] =
temp.iloc[225,5-a]
                        UPDRS[name, "v5_4_right_rest_tremor_constancy"] = temp.iloc[226,5-a]

                        UPDRS[name, "v5_4_left_postural_tremor"] = temp.iloc[222,7-a]
                        UPDRS[name, "v5_4_left_kinetic_tremor"] = temp.iloc[223,7-a]
```

```
                        UPDRS[name, "v5_4_left_rest_tremor_amplitude_arm"] = temp.iloc[224,7-a]
                        UPDRS[name, "v5_4_left_rest_tremor_amplitude_leg"] = temp.iloc[225,7-a]
                        UPDRS[name, "v5_4_left_rest_tremor_constancy"] = temp.iloc[226,7-a]

                        UPDRS[name, "v5_4_list_rig_tot"] =
(temp.iloc[201,5-a],temp.iloc[202,5-a],temp.iloc[203,5-a],temp.iloc[202
,7-a],temp.iloc[203,7-a])
                        UPDRS[name, "v5_4_list_speech_facial"] =
(temp.iloc[205,5-a],temp.iloc[206,5-a])
                        UPDRS[name, "v5_4_list_brady_kin_right"] = (temp.iloc[208,5-a],
temp.iloc[209,5-a], temp.iloc[210,5-a], temp.iloc[211,5-a],
temp.iloc[212,5-a])
                        UPDRS[name, "v5_4_list_brady_kin_left"] = (temp.iloc[208,7-a],
temp.iloc[209,7-a], temp.iloc[210,7-a], temp.iloc[211,7-a],
temp.iloc[212,7-a])
                        UPDRS[name, "v5_4_list_brady_kin_tot"] = (UPDRS[name,
"v5_4_list_brady_kin_left"] + UPDRS[name, "v5_4_list_brady_kin_right"])
                        UPDRS[name, "v5_4_list_gait_balance"] = (temp.iloc[214,5-a],
temp.iloc[215,5-a], temp.iloc[216,5-a], temp.iloc[217,5-a],
temp.iloc[218,5-a], temp.iloc[219,5-a])
                        UPDRS[name, "v5_4_list_tremor_right"]= (temp.iloc[222,5-a],
temp.iloc[223,5-a], temp.iloc[224,5-a], temp.iloc[225,5-a],
temp.iloc[226,5-a])
                        UPDRS[name, "v5_4_list_tremor_left"]= (temp.iloc[222,7-a],
temp.iloc[223,7-a], temp.iloc[224,7-a], temp.iloc[225,7-a],
temp.iloc[226,7-a] )
                        UPDRS[name, "v5_4_list_tremor_tot"]= (UPDRS[name,
"v5_4_list_tremor_right"] + UPDRS[name, "v5_4_list_tremor_left"])
                        UPDRS[name, "v5_4_list_globaal"] = (UPDRS[name, "v5_4_list_rig_tot"] +
UPDRS[name, "v5_4_list_speech_facial"] + UPDRS[name,
"v5_4_list_brady_kin_tot"] + UPDRS[name, "v5_4_list_gait_balance"] +
UPDRS[name, "v5_4_list_tremor_tot"])
UPDRS_dataframe = pd.DataFrame.from_dict(UPDRS, orient='index')
UPDRS_dataframe.to_excel(r"F:\Datasets\UPDRS_scores_filtered.xlsx")
#%%
list_keys = sorted(UPDRS.keys())
retrieved_data = {}
for x in range(len(list_keys[0:300])): #change the numbers for which values you want to
extract from the UPDRS table
```

```
    temp2=list_keys[x]
    retrieved_data[list_keys[x]] = UPDRS[temp2]
#%% Plot tremor total
fig = plt.figure()

#define the x-labels and the y-labels
Xlabel =
["right_postural_tremor","right_kinetic_tremor","right_rest_tremor_amplitude_arm","right_rest_t
remor_amplitude_leg","right_rest_tremor_constancy","left_postural_tremor","left_kinetic_tremor"
,"left_rest_tremor_amplitude_arm","left_rest_tremor_amplitude_leg","left_rest_tremor_constancy"
]
Blabel = ["Visit 2", "Visit 3", "Visit 4", "Visit 5 analysis 1", "Visit 5 analysis 2", "Visit
5 analysis 3", "Visit 5 analysis 4"]
data_plot = [0,1,2,3,4,5,6,7,8,9] #maak een list die 10 lang is

#fill the aforementioned list with the correct values.
for x in range(len(UPDRS["ptxx", "v2_list_tremor_tot"])):
    data_plot[x]=[int(UPDRS["ptxx", "v2_list_tremor_tot"][x]),int(UPDRS["pt0",
"v3_list_tremor_tot"][x]),int(UPDRS["ptxx", "v4_list_tremor_tot"][x]),int(UPDRS["ptxx",
"v5_1_list_tremor_tot"][x]),int(UPDRS["ptxx", "v5_2_list_tremor_tot"][x]),int(UPDRS["ptxx",
"v5_3_list_tremor_tot"][x]),int(UPDRS["ptxx", "v5_4_list_tremor_tot"][x])]

#create the bar charts and the distance between the axes:
x0=np.arange(len(data_plot[0]))
x1=[x+0.1 for x in x0]
x2=[x+0.1 for x in x1]
x3=[x+0.1 for x in x2]
x4=[x+0.1 for x in x3]
x5=[x+0.1 for x in x4]
x6=[x+0.1 for x in x5]
x7=[x+0.1 for x in x6]
x8=[x+0.1 for x in x7]
x9=[x+0.1 for x in x8]

plt.subplots(figsize=[16,9])
plt.bar(x0, data_plot[0], color = 'b', width = 0.1,label = Xlabel[0])
plt.bar(x1, data_plot[1], color = 'g', width = 0.1,label = Xlabel[1])
plt.bar(x2, data_plot[2], color = 'r', width = 0.1,label = Xlabel[2])
plt.bar(x3, data_plot[3], color = 'c', width = 0.1,label = Xlabel[3])
```

```
plt.bar(x4, data_plot[4], color = 'm', width = 0.1,label = Xlabel[4])
plt.bar(x5, data_plot[5], color = 'y', width = 0.1,label = Xlabel[5])
plt.bar(x6, data_plot[6], color = 'k', width = 0.1,label = Xlabel[6])
plt.bar(x7, data_plot[7], color = 'gray', width = 0.1,label = Xlabel[7])
plt.bar(x8, data_plot[8], color = 'pink', width = 0.1,label = Xlabel[8])
plt.bar(x9, data_plot[9], color = 'purple', width = 0.1,label = Xlabel[9])

#adds labels to graphic
plt.xlabel('Visit & Test', fontweight ='bold', fontsize = 20)
plt.ylabel('UPDRS-score', fontweight ='bold', fontsize = 20)
plt.xticks([r + 0.1 for r in range(len(data_plot[0]))], Blabel, fontsize = 12)
plt.title("Patients tremor UPDRS-scores across multiple visits", fontweight = "bold", fontsize
= 30)

plt.legend(fontsize = 8)
plt.show()
```

8.2.8 05_pre_algorithm

```
    # -*- coding: utf-8 -*-
"""
Created on Tue May 30 11:44:18 2023

@author: jobme
"""
import io
import os
import datetime as dt
import time
import pandas as pd
import matplotlib.pyplot as plt
from runeq import Config, stream
from scipy import stats
import numpy
#this script acquires an excell table with smartwatchdata during examination of MDS-UPDRS
#%%
#load file from patient
BASE_PATH = 'G:\Datasets'
PT_FILENAME = '\ptxx_wtxx' #sometimes you need to adjust the wt03 higher (wt04) or lower(wt02)
recieve= BASE_PATH+PT_FILENAME+PT_FILENAME
```

```
store = BASE_PATH+PT_FILENAME+PT_FILENAME
#%%
df1 = pd.read_excel(recieve+'_merged.xlsx')



#%%
#choose date and time interval of examination
date_time = '29.11.2022 10:00:00'
pattern = '%d.%m.%Y %H:%M:%S'
epoch = int(time.mktime(time.strptime(date_time, pattern)))
#print (epoch)

#%% time window
end_time= epoch + 3600 #hour past measurement
start_time=epoch - 900 #fifteen minutes before measurement
#%%

df5 = df1[(df1['time'] > start_time) & (df1['time'] < end_time)]
df6 = numpy.mean(df5) #get the mean value of each parameter during the time interval

#Added
df8=df6[2:9]

#%%
#wb = openpyxl.Workbook()
#ws = wb.active
#in format
("mean_tremor","mild","moderate","none","slight","strong","unknown","mean_dyskinesia"),
#data = (
("mean_tremor","mild","moderate","none","slight","strong","unknown","mean_dyskinesia"),
(df8[0],df8[1],df8[2],df8[3],df8[4],df8[5],df8[6]),
)
#%%
#for i in data:
   ws.append(i)
#%%
#ws.append(i)
#wb.save('G:\data\output.xlsx') #file is save as
```

8.2.9 06_merge_UPDRS_loadr_data_and_smartwatch_data

```
    # -*- coding: utf-8 -*-
"""
Data_final_conversion

Purpose: combines scripts retrieved from the UPDRS_loadr script and the table with smartwatch
averages to one
table that can be easily incorporated into a ML-algorithm

Uses pandas v. 2.0.1

Created on Tue May 30 12:11 2023, last edit: Thu June 1 14:08 2023

@authors: Loet Schoenmakers, Job Meuwese & Stef Berendsen
"""

#%% import nessecery libraries
import pandas as pd

#%% Paths to specify by the user:
path_UPDRS = r"F:\Datasets\UPDRS_scores_filtered.xlsx" #the location of the UPDRS_loadr
excel-file
path_pt_summary_data = r"F:\Datasets\pre_algorithm.xlsx" #the location of the smartwatch
averages excel-file
path_pt_data_location = r"F:\Datasets\data_complete.xlsx" #the location you wish the new table
to end up at

#%% load UPDRS- and Smartwatch-data (Variables changeable by user)

list_of_pt = [] #specify which pt's you
wish to include in the table
visits = ["v2_","v3_","v4_","v5_1_","v5_2_","v5_3_","v5_4_"] #specify which visits you wish to
include in the table. DO NOT FORGET THE "_" (LOWER DASH) AT THE END OF THE VISIT NAME!

#below here, specify what labels of the UPDRS_loadr and Smartwatch data scripts you wish to
include in the table. DO NOT INCLUDE ANY TRAILING OR LEADING BLANK SPACES IN THESE LABELS!
list_UPDRS_labels =
["right_finger_tap","right_hand_movement","right_pro_supination","right_toe_tap","right_leg_mov
ement","left_finger_tap","left_hand_movement","left_pro_supination","left_toe_tap","left_leg_mo
```

```
vement","raise_chair","gait","freezing","postural_reflex","posture","spontaneous_move","right_p
ostural_tremor",
"right_kinetic_tremor","right_rest_tremor_amplitude_arm","right_rest_tremor_amplitude_leg","rig
ht_rest_tremor_constancy","left_postural_tremor",
"left_kinetic_tremor","left_rest_tremor_amplitude_arm","left_rest_tremor_amplitude_leg","left_r
est_tremor_constancy"]
list_smartwatch_labels =
["mean_tremor","mean_mild","mean_moderate","mean_none","mean_slight","mean_strong","mean_unknow
n","mean_dyskinesia"]

#%% Load UPDRS- and Smartwatch data (unchangeable)
#pre-define needed variables for later on
List_labels_tot = list_UPDRS_labels + list_smartwatch_labels
index_labels = {}
pt_names = list_of_pt

#the next six lines load the UPDRS_loadr script, and remove any trailing and leading spaces
from the labels it may contain.
UPDRS_data = pd.read_excel(path_UPDRS)
UPDRS_data = UPDRS_data.rename({"Unnamed: 0":"names",0:"data"}, axis="columns")

UPDRS_data_labels = UPDRS_data.names
UPDRS_data_labels = UPDRS_data_labels.tolist()
UPDRS_data_labels = [s.strip() for s in UPDRS_data_labels]

#make a list of pt-names in the format "ptXX", where XX are the numbers in the list_of_pt
list. post these labels in the pt_names list. then (after line 52), combine it with the visits
specified in the index_labels dict
for x in range(len(list_of_pt)):
    if list_of_pt[x] >=10:
        pt_names[x]="pt"+str(list_of_pt[x]).strip() #define the name
    else:
        pt_names[x]="pt0"+str(list_of_pt[x]).strip()
    for y in range(len(visits)):
        index_labels[x,y] = pt_names[x]+"_visit"+visits[y][1:-1]

#define the pt_data dataframe, using the rows and colums made earlier.
pt_data = pd.DataFrame(index = index_labels.values(),columns=List_labels_tot)
```

```
#load in the smartwatch data, and remove any trailing or l;eading spaces the labels may
contain.
pt_summary_data = pd.read_excel(path_pt_summary_data, usecols="A:K")
pt_summary_data = pt_summary_data.dropna().drop(columns = "time_window").reset_index()
pt_summary_columns = list(pt_summary_data.columns)
pt_summary_columns = pt_summary_columns[3:]
pt_summary_columns = [s.strip() for s in pt_summary_columns]

#make a list of pt-names, combined with visits that are present in the smartwatch-data excel
sheet.
pt_summary_list = [None]*len(pt_summary_data.number_patient)
for z in range(len(pt_summary_data.number_patient)):
    pt_summary_list[z] =
str(pt_summary_data.number_patient[z])+"_visit"+str(pt_summary_data.number_of_visit[z]).str
ip()
#%% fill in the patient data DataFrame

"""
from line 78 to 93, the script does the following operations for each row of the empty table
(line 78):
    - make 3 temporary variables that are empty (line 79)
    - check if the patient (line 80), visit (lines 82 & 86) and UPDRS-score category (lines 84
& 88) are present in the list_of_pt, visits, and list_UPDRS_lables lists defined earlier.
if so, temporarily store these in temp 1, 2, and 3 respectively
    - if all 3 labels (pt, visit and UPDRS_score_category) are present (line 90), make the
index-label of the row in the pt_data DataFrame and strip any blank spaces (lines 91 &
92), then place it in the cooresponding field (line 93)
"""

for a in range(len(UPDRS_data_labels)):
    temp1 = temp2 = temp3 = None
    if UPDRS_data_labels[a][2:6] in pt_names:
        temp1 = UPDRS_data_labels[a][2:6]
        if UPDRS_data_labels[a][10:13] in visits[:3]:
            temp2 = UPDRS_data_labels[a][11:12]
            if UPDRS_data_labels[a][13:-2] in list_UPDRS_labels:
                temp3 = UPDRS_data_labels[a][13:-2]
        elif UPDRS_data_labels[a][10:15] in visits[3:]:
            temp2 = UPDRS_data_labels[a][11:14]
```

```
            if UPDRS_data_labels[a][15:-2] in list_UPDRS_labels:
                temp3 = UPDRS_data_labels[a][15:-2]
    if temp1 and temp2 and temp3 != None:
        temp_label = temp1+"_visit"+temp2
        temp_label = temp_label.strip()
        pt_data.loc[[temp_label],[temp3]] = UPDRS_data.data[a]

"""
from line 102 to 114, the script does the following operations for each row of the smartwatch
data sheet (line 102):
    - make 3 temporary variables that are empty (line 103)
    - check if the patient (line 104), visit (line 104) and smartwatch data categories (lines
108 & 109) are present in the list_of_pt, visits, and list_smartwatch_labels lists defined
earlier. if so, temporarily store these in temp 4, 5 and 6 respectively
    - if all 3 labels (pt, visit and UPDRS_score_category) are present (line 111), make the
index-label of the row in the pt_data DataFrame and strip any blank spaces (lines 112 &
113), then place it in the cooresponding field (line 114)
"""

for b in range(len(pt_summary_list)):
     temp4 = temp5 = temp6 = None
     if pt_summary_list[b][:4] in pt_names:
         temp4 = pt_summary_list[b][:4]
         if pt_summary_list[b][5] + pt_summary_list[b][10:] + "_" in visits:
             temp5 = pt_summary_list[b][10:]
             for c in range(len(pt_summary_columns)):
                 if pt_summary_columns[c] in list_smartwatch_labels:
                     temp6 = pt_summary_columns[c]
                     if temp4 and temp5 and temp6 != None:
                         temp_label2 = temp4+"_visit"+temp5
                         temp_label2 = temp_label2.strip()
                         pt_data.loc[[temp_label2],[temp6]] = pt_summary_data.at[b,temp6]

#%% Post-processing and export of the complete table: (can be run
optionally)
pt_data = pt_data.dropna()   #remove any row, which does not contain a full set of data. Can
optionally be commented.
pt_data.to_excel(path_pt_data_location) #export to specific file location.
```

8.2.10 07_1_algorithm01

```
    # -*- coding: utf-8 -*-
"""
Created on Wed May 31 09:44:56 2023

@author: Stef Berendsen, Loet Schoenmakers en Job Meuwese

Purpose: to create an RF-algorithm for retrieving UPDRS-scores from smartwatch data

Makes use of the following libraries:
    scikit-learn v: 1.2.2
    pandas v: 2.0.1
    matplotlib v: 3.7.1

Also makes use of the following data-set:
    data_complete.xlsx
retrievable from the script "06_merge_UPDRS_loadr_data_and_smartwatch_data". This script is
available on the GitHub
"""

#%% Import the nescesary libraries:
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt

#%% Import the files and define variables:
#please specify within the '' the path of the data_complete.xlsx file.
df= pd.read_excel('E:/data/data_complete.xlsx',index_col=0)

#df.drop means that these columms will be removed so you only have the ones you desire
X =
df.drop(["right_finger_tap","right_hand_movement","right_pro_supination","right_toe_tap","right
_leg_movement","left_finger_tap","left_hand_movement","left_pro_supination","left_toe_tap","lef
t_leg_movement","raise_chair","gait","freezing","postural_reflex","posture","spontaneous_move",
"right_postural_tremor",
"right_kinetic_tremor","right_rest_tremor_amplitude_arm","right_rest_tremor_amplitude_leg","rig
ht_rest_tremor_constancy","left_postural_tremor",
"left_kinetic_tremor","left_rest_tremor_amplitude_arm","left_rest_tremor_amplitude_leg","left_r
```

```
est_tremor_constancy"], axis=1)
y =
df.drop(["mean_tremor","mean_mild","mean_moderate","mean_none","mean_slight","mean_strong","mea
n_unknown","mean_dyskinesia"],axis=1)

#%% Algorithm
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=None)
#train is used the train the system, test is to use the system, test_size 0.2 says that 20%
percent of the values wil be used to test the system and 80% to train

rf_model = RandomForestClassifier(n_estimators=100, max_features="auto", random_state=None)
#n_estimators gives the amount of decision trees,max_features deletes the worst values if it
is contradicted
rf_model.fit(X_train, y_train) #fit the data to the algorithm

predictions = rf_model.predict(X_test)# gives the predicitons for the whole tree, through mean
y_test

propbabilitys=rf_model.predict_proba(X_test) #for each field of columm gives the probabilitys
of y_values if it is 1,2,3,4
importances=rf_model.feature_importances_ #how important the columms in X_test are in
predicting results

#%% Add random person and check it's predictions.
person_test=[[0.025, 0.5, 0.00345, 0.2655, 0.6902, 0.12345, 0.3874, 0.4999]]
person_result=rf_model.predict(person_test) # gives the predict values of updrs with the
random smartwatchdata
```

8.2.11 07_2_algorithm02

```
    # -*- coding: utf-8 -*-
"""
Created on Mon Jun  5 13:36:45 2023

@author: Stef Berendsen, Loet Schoenmakers en Job Meuwese

Purpose: to create a more optimised RF-algorithm for retrieving UPDRS-scores from smartwatch
data

Makes use of the following libraries:
```

```
    scikit-learn v: 1.2.2
    pandas v: 2.0.1

Also makes use of the following data-set:
    data_complete.xlsx
retrievable from the script "06_merge_UPDRS_loadr_data_and_smartwatch_data". This script is
available on the GitHub
"""

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pandas as pd

#%%
#specify the data_complete .xlsx path within the ''.
df= pd.read_excel('F:/Datasets/data_complete.xlsx',index_col=0)

#define the labels of the UPDRS- and Smartwatch data
UPDRS_labels =
["right_finger_tap","right_hand_movement","right_pro_supination","right_toe_tap","right_leg_mov
ement","left_finger_tap","left_hand_movement","left_pro_supination","left_toe_tap","left_leg_mo
vement","raise_chair","gait","freezing","postural_reflex","posture","spontaneous_move","right_p
ostural_tremor",
"right_kinetic_tremor","right_rest_tremor_amplitude_arm","right_rest_tremor_amplitude_leg","rig
ht_rest_tremor_constancy","left_postural_tremor",
"left_kinetic_tremor","left_rest_tremor_amplitude_arm","left_rest_tremor_amplitude_leg","left_r
est_tremor_constancy"]
smartwatch_labels =
["mean_tremor","mean_mild","mean_moderate","mean_none","mean_slight","mean_strong","mean_unknow
n","mean_dyskinesia"]

#df.drop means that these columms will be removed so you only have the ones you desire
X = df.drop(UPDRS_labels, axis=1)
y = df.drop(smartwatch_labels,axis=1)

#define the nescerary variables to store data of the RF in
X_train = {}
y_train = {}
X_test = {}
```

```
y_test = {}
rf_model = {}
predictions = {}

for a in range(len(UPDRS_labels)): #for each of the UPDRS-labels:

    #train is used to train the system, test is to test the system, test_size = 0.2 means that
20% percent of the values wil be used to test the system and 80% to train the system.
    X_train[a], X_test[a], y_train[a], y_test[a] = train_test_split(X, y[UPDRS_labels[a]],
test_size=0.2, random_state=None) #train is used the train the system, test is to use the
system, test_size 0.2 says that 20% percent of the values wil be used to test the system
and 80% to train

    #create a RF for that UPDRS-score, fit it with the correct data, and make predictions
based from it.
    rf_model[a] = RandomForestClassifier(n_estimators=100, max_features="sqrt",
random_state=None) #n_estimators gives the amount of decision trees,max_features deletes
the worst values if it is contradicted
    rf_model[a].fit(X_train[a], y_train[a])
    predictions[a] = rf_model[a].predict(X_test[a])

#%%
# gives the predicitons for the whole tree, through mean y_test, first, define nescesary
variables
accuracy = [None]*len(UPDRS_labels)
accuracy_within_one = [None]*len(UPDRS_labels)
for z in range(len(UPDRS_labels)): #for each UPDRS-label:

    #define temporary variables
    num_spot_on = 0
    num_within_one = 0
    total = 0
    for b in range(len(y_test[z].index.to_list())): #for each test patient in run z:
        if predictions[z][b] == y_test[z][b]: #if the predicted number matches exactly with
the test data:
            num_spot_on = num_spot_on + 1
            total = total + 1
        elif y_test[z][b] - 1 <= predictions[z][b] <= y_test[z][b] + 1: #if the predicted
number is within-one of the test data
```

```
            num_within_one = num_within_one + 1
            total = total + 1
        else: #if the predicted number does not match any of the requirements above:
            total = total + 1

    num_within_one = num_spot_on + num_within_one #add the total of the exact numbers to the
within-one numbers

    #calculate the accuracies for run z:
    accuracy[z] = num_spot_on/total*100
    accuracy_within_one[z] = num_within_one/total*100

#calculate the accuracies over all runs:
accuracy_tot = sum(accuracy) / len(accuracy)
accuracy_within_one_tot = sum(accuracy_within_one) / len(accuracy_within_one)
```

8.2.12 07_3_algorithm03.py

```
    # -*- coding: utf-8 -*-
"""
Created on Fri Jun  9 13:49:50 2023

@author: Stef Berendsen, Loet Schoenmakers en Job Meuwese (B-TG students @ University of
Twente)

Purpose: to create the most optimised RF-algorithm for retrieving UPDRS-scores from smartwatch
data

Makes use of the following libraries:
    scikit-learn v: 1.2.2
    pandas v: 2.0.1
    numpy v: 1.24.3

Also makes use of the following data-set:
    data_complete.xlsx
retrievable from the script "06_merge_UPDRS_loadr_data_and_smartwatch_data". This script is
available on the GitHub
"""

#%%Import Libraries
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.metrics import precision_score, recall_score
import pandas as pd
import numpy as np

#%% Load in data, filter data in UPDRS and Smartwatch data.
#specify the data_complete .xlsx path within the ''.
df= pd.read_excel('F:/Datasets/data_complete.xlsx',index_col=0)

#define the labels of the UPDRS- and Smartwatch data
UPDRS_labels =
["right_finger_tap","right_hand_movement","right_pro_supination","right_toe_tap","right_leg_mov
ement","left_finger_tap","left_hand_movement","left_pro_supination","left_toe_tap","left_leg_mo
vement","raise_chair","gait","freezing","postural_reflex","posture","spontaneous_move","right_p
ostural_tremor",
"right_kinetic_tremor","right_rest_tremor_amplitude_arm","right_rest_tremor_amplitude_leg","rig
ht_rest_tremor_constancy","left_postural_tremor",
"left_kinetic_tremor","left_rest_tremor_amplitude_arm","left_rest_tremor_amplitude_leg","left_r
est_tremor_constancy"]
smartwatch_labels =
["mean_tremor","mean_mild","mean_moderate","mean_none","mean_slight","mean_strong","mean_unknow
n","mean_dyskinesia"]

#df.drop means that these columms will be removed so you only have the ones you desire
X = df.drop(UPDRS_labels, axis=1) #creates smartwatch-data list
y = df.drop(smartwatch_labels,axis=1)  #creates the UPDRS-list

#define the number of RF's you wish to make in num_of_runs
num_of_runs = 50

#make lists and a dict to store the respective variables in
accuracy =  [None]*num_of_runs #the % of predicted points that directly match the test data
("one-on-one accuracy")
accuracy_within_one = [None]*num_of_runs #the % of predicted points that are within one of the
respective test data ("within-one accuracy")
precision = [None]*num_of_runs #average of: tp / (tp + fp) (tp = true positive, fp = false
positive. Dutch: de positieve predictieve waarde)
```

```
recall =[None]*num_of_runs #average of: tp / (tp + fn) (tp = true positive, fn = false
negative. Dutch: de sensitiviteit)
rf_model = {}

#%% Runs the RF a number of times equal to num_of_runs
for a in range(num_of_runs):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=None) #train is used to train the system, test is to test the system,
test_size = 0.2 means that 20% percent of the values wil be used to test the system and
80% to train the system.

    param_dict = {'n_estimators': np.arange(100, 450, 10), # The number of decision trees used
to make a random forest. varies between 100 and 450 with increments of 10
                'max_depth': np.array([1,25,30,35,50,55,60,65,70,75,80,85]), #the most
branches the decision tree will go down before finishing. See the numbers in
the list.
                "bootstrap": [True, False], #If bootstrap = true, use random samples from
train set to train individual decision trees. If bootstrap = false, use all
samples instead.
                'min_samples_leaf': [1,2,3,4,5], #the minimum number of samples needed to
cause a node to be a leaf nnode (the last one on that line)
                'min_samples_split': [15, 18, 20, 21, 22, 23, 24, 25, 30], #the minimum
number of samples needed to create a split in the decision tree
                "warm_start": [True,False]} #if warm_start = True, it reuses the previous
call to add more samples to the RF. If warm_start = False, create a new RF
instead.

    # Create a random forest classifier
    rf = RandomForestClassifier()

    # Use random search to find the best hyperparameters in the param_dict variable. choose
n_iter number of variables randomly and use a 5-fold cross-validation (cv = 5) to check
this data.
    #For users: n_jobs = -1 makes it so that all CPU cores are used for the RF-algorithm (this
makes for the fastest computing times possible, but also slows down other programs a lot).
if you want a slower RF, but one that does not slow down your PC as much, use n_jobs = 1
or a smaller, whole number than -1 instead.
    rf_model[a] = RandomizedSearchCV(rf, param_distributions = param_dict, n_iter=25, cv=5,
n_jobs=-1)
```

```
    # Fit the parameters of the random search to the training data
    rf_model[a].fit(X_train, y_train)

    # gives the predicitons for the whole tree, through mean y_test
    y_pred = rf_model[a].predict(X_test)

    #prepare labels to make calculating the accuracy easier.
    index_predictions = y_test.index
    index_predictions2 = y_test.index.to_list()
    predictions = pd.DataFrame(y_pred, columns=UPDRS_labels)
    predictions = predictions.set_index(index_predictions)

    #prepare variables to keep track of the number of exact measures and of the measures
within-one
    num_spot_on = 0
    num_within_one = 0
    total = 0
    for c in index_predictions2: #for each patient in the test data:
        for d in UPDRS_labels: #for each UPDRS-label
            if predictions.at[c,d] == y_test.at[c,d]: #if the predicted data is an exact match
with the test data on the same location:
                num_spot_on = num_spot_on + 1
                total = total + 1
            elif y_test.at[c,d] - 1 <= predictions.at[c,d] <= y_test.at[c,d] + 1:  #if the
predicted data is within one of the test data on the same location:
                num_within_one = num_within_one + 1
                total = total + 1
            else:  #if the data deviates more than one:
                total = total + 1

    num_within_one = num_spot_on + num_within_one #add the number of exact matches to the
within-one data

    accuracy[a] = num_spot_on/total*100 #calculate the accuracy (one-on-one) of that run
    accuracy_within_one[a] = num_within_one/total*100 #calculate the accuracy (within-one) of
that run

    #calculate the precision and recall values of the algorithm.
```

```
    pris_temp = rec_temp = [None]*len(index_predictions2)
    for b in range(len(index_predictions2)): #for each patient
        pris_temp[b] =
precision_score(y_test.loc[index_predictions2[b]].reset_index(drop=True).to_list(),
predictions.loc[index_predictions2[b]].reset_index(drop=True).to_list(),
average='macro', zero_division=1)
        rec_temp[b] =
recall_score(y_test.loc[index_predictions2[b]].reset_index(drop=True).to_list(),
predictions.loc[index_predictions2[b]].reset_index(drop=True).to_list(),
average='macro', zero_division=1)
    precision[a] = sum(pris_temp)/len(pris_temp)
    recall[a] = sum(rec_temp)/len(rec_temp)
    #propbabilitys=rf_model.predict_proba(X_test) #for each field of columm gives the
probabilitys of y_values if it is 1,2,3,4
    #importances=rf_model.feature_importances_ #how important the columms in X_test are in
predicting results

#give the average accuracies over the number of RF's
accuracy_avg = sum(accuracy)/len(accuracy)
accuracy_within_one_avg = sum(accuracy_within_one)/len(accuracy_within_one)
```

8.2.13 08_Analyse_complete_data_tabel

```
    # -*- coding: utf-8 -*-
"""
Created on Thu Jun  1 11:44:12 2023

@author: Loet Schoenmakers, Stef Berendsen en Job Meuwese
"""
import pandas as pd
import matplotlib.pyplot as plt
import math
from scipy import stats

df = pd.read_excel('G:/data/data_complete.xlsx',index_col=0) #output of 06 on github is input
for these analyses
#%% input visit
patient_number='03'#two digits thus 08 for example
patient_visit='2' #single digit for visit 5 '5_x'
pt_visit = 'pt'+patient_number+'_visit'+patient_visit
```

```
#%% input comparison
patient_ctrl_number='xx'#two digits thus 08 for example
patient_ctrl_visit='x' #single digit for visit 5 '5_x'
pt__crtl_visit = 'pt'+patient_ctrl_number+'_visit'+patient_ctrl_visit

#%% visit UPDRS related sums up each test to a certain category
int_row = df[df.index == pt_visit] #row (pt) of interest

v_brady_right=sum(int_row['right_finger_tap']+int_row['right_hand_movement']+int_row['right_pro
_supination']+int_row['right_toe_tap']+int_row['right_leg_movement'])
v_brady_left=sum(int_row['left_finger_tap']+int_row['left_hand_movement']+int_row['left_pro_sup
ination']+int_row['left_toe_tap']+int_row['left_leg_movement'])

v_trem_right=sum(int_row['right_postural_tremor']+int_row['right_kinetic_tremor']+int_row['righ
t_rest_tremor_amplitude_arm']+int_row['right_rest_tremor_amplitude_leg']+int_row['right_rest_tr
emor_constancy'])
v_trem_left=sum(int_row['left_postural_tremor']+int_row['left_kinetic_tremor']+int_row['left_re
st_tremor_amplitude_arm']+int_row['left_rest_tremor_amplitude_leg']+int_row['left_rest_tremor_c
onstancy'])

v_gait_balance=sum(int_row['raise_chair']+int_row['gait']+int_row['freezing']+int_row['postural
_reflex']+int_row['posture']+int_row['spontaneous_move'])

#%% visit smartwatch related
v_mean_tremor=sum(int_row['mean_tremor'])
v_mean_mild=sum(int_row['mean_mild'])
v_mean_moderate=sum(int_row['mean_moderate'])
v_mean_none=sum(int_row['mean_none'])
v_mean_slight=sum(int_row['mean_slight'])
v_mean_strong=sum(int_row['mean_strong'])
v_mean_unknown=sum(int_row['mean_unknown'])
v_mean_dyskinesia=sum(int_row['mean_dyskinesia'])

#%% comparison UPDRS related
int_row_c = df[df.index == pt__crtl_visit] #row (pt) of interest

c_brady_right=sum(int_row_c['right_finger_tap']+int_row_c['right_hand_movement']+int_row_c['rig
ht_pro_supination']+int_row_c['right_toe_tap']+int_row_c['right_leg_movement'])
```

```
c_brady_left=sum(int_row_c['left_finger_tap']+int_row_c['left_hand_movement']+int_row_c['left_p
ro_supination']+int_row_c['left_toe_tap']+int_row_c['left_leg_movement'])

c_trem_right=sum(int_row_c['right_postural_tremor']+int_row_c['right_kinetic_tremor']+int_row_c
['right_rest_tremor_amplitude_arm']+int_row_c['right_rest_tremor_amplitude_leg']+int_row_c['rig
ht_rest_tremor_constancy'])
c_trem_left=sum(int_row_c['left_postural_tremor']+int_row_c['left_kinetic_tremor']+int_row_c['l
eft_rest_tremor_amplitude_arm']+int_row_c['left_rest_tremor_amplitude_leg']+int_row_c['left_res
t_tremor_constancy'])

c_gait_balance=sum(int_row_c['raise_chair']+int_row_c['gait']+int_row_c['freezing']+int_row_c['
postural_reflex']+int_row_c['posture']+int_row_c['spontaneous_move'])


#%% comparisom smartwatch related
c_mean_tremor=sum(int_row_c['mean_tremor'])
c_mean_mild=sum(int_row_c['mean_mild'])
c_mean_moderate=sum(int_row_c['mean_moderate'])
c_mean_none=sum(int_row_c['mean_none'])
c_mean_slight=sum(int_row_c['mean_slight'])
c_mean_strong=sum(int_row_c['mean_strong'])
c_mean_unknown=sum(int_row_c['mean_unknown'])
c_mean_dyskinesia=sum(int_row_c['mean_dyskinesia'])


#%% availability check
check=[v_trem_right,c_trem_right,v_mean_tremor,c_mean_tremor] # checks availability of most
important value for each UPDRS-visit or smartwatchdata
for x in check:
    check_trem=math.isnan(x)
    if check_trem == False:
        print('available')
    if check_trem == True:
        check_trem=str(check_trem)
        print('is not available')


#%% creates storage for input plots
n = 38
```

```
len_plot = []

for i in range(0, n+1):
    len_plot.append(i)

data_plot = len_plot #maak een list die 10 lang is
#%% fills in plotted info
data_plot[0]= v_trem_right
data_plot[1]= v_trem_left
data_plot[2]= v_brady_right
data_plot[3]= v_brady_left
data_plot[4]= v_gait_balance

data_plot[5]= c_trem_right
data_plot[6]= c_trem_left
data_plot[7]= c_brady_right
data_plot[8]= c_brady_left
data_plot[9]= c_gait_balance

data_plot[10]= v_mean_tremor
data_plot[11]= v_mean_mild
data_plot[12]= v_mean_moderate
data_plot[13]= v_mean_none
data_plot[14]= v_mean_slight
data_plot[15]= v_mean_strong
data_plot[16]= v_mean_unknown
data_plot[17]= v_mean_dyskinesia

data_plot[18]=c_mean_tremor
data_plot[19]=c_mean_mild
data_plot[20]=c_mean_moderate
data_plot[21]=c_mean_none
data_plot[22]=c_mean_slight
data_plot[23]=c_mean_strong
data_plot[24]=c_mean_unknown
data_plot[25]=c_mean_dyskinesia

#%%
data_plot[26]= c_trem_right-v_trem_right
```

```
data_plot[27]= c_trem_left - v_trem_left
data_plot[28]= c_brady_right - v_brady_right
data_plot[29]= c_brady_left - v_brady_left
data_plot[30]= c_gait_balance - v_gait_balance

data_plot[31]=c_mean_tremor - v_mean_tremor
data_plot[32]=c_mean_mild - v_mean_mild
data_plot[33]=c_mean_moderate - v_mean_moderate
data_plot[34]=c_mean_none - v_mean_none
data_plot[35]=c_mean_slight - v_mean_slight
data_plot[36]=c_mean_strong - v_mean_strong
data_plot[37]=c_mean_unknown - v_mean_unknown
data_plot[38]=c_mean_dyskinesia - v_mean_dyskinesia

Xlabel = ["v_trem_right","v_trem_left","v_brady_right","v_brady_left","v_gait_balance",
          "c_trem_right","c_trem_left","c_brady_right","c_brady_left","c_gait_balance",
          """v_mean_
          tremor""","""v_mean_
          mild""","""v_mean_
          moderate""","""v_mean_
          none""",
          """v_mean_
          slight""","""v_mean_
          strong""","""v_mean_
          unknown""","""v_mean_
          dyskinesia""",
          """c_mean_
          tremor""","""c_mean_
          mild""","""c_mean_
          moderate""","""c_mean_
          none""","""c_mean_
          slight""","""c_mean_
          strong""","""c_mean_
          unknown""","""c_mean_
          dyskinesia""",

'dif_trem_right','dif_trem_left','dif_brady_right','dif_brady_left','dif_gait_balance
',
          """dif_mean_
```

```
          tremor""","""dif_mean_
          mild""","""dif_mean_
          moderate""","""dif_mean_
          none""",
          """dif_mean_
          slight""","""dif_mean_
          strong""","""dif_mean_
          unknown""","""dif_mean_
dyskinesia"""
          ]

#%% plotting the values
fig, ((ax1,ax2),(ax3,ax4),(ax5,ax6))=plt.subplots(3, 2,figsize=[24,16])
ax1.bar(Xlabel[0], data_plot[0], color = '#104E8B')
ax1.bar(Xlabel[1], data_plot[1], color = '#1E90FF')
ax1.bar(Xlabel[2], data_plot[2], color = '#104E8B')
ax1.bar(Xlabel[3], data_plot[3], color = '#1E90FF')
ax1.bar(Xlabel[4], data_plot[4], color = '#104E8B')

ax3.bar(Xlabel[5], data_plot[5], color = '#104E8B')
ax3.bar(Xlabel[6], data_plot[6], color = '#1E90FF')
ax3.bar(Xlabel[7], data_plot[7], color = '#104E8B')
ax3.bar(Xlabel[8], data_plot[8], color = '#1E90FF')
ax3.bar(Xlabel[9], data_plot[9], color = '#104E8B')

ax2.bar(Xlabel[10], data_plot[10], color = '#104E8B')
ax2.bar(Xlabel[11], data_plot[11], color = '#1E90FF')
ax2.bar(Xlabel[12], data_plot[12], color = '#104E8B')
ax2.bar(Xlabel[13], data_plot[13], color = '#1E90FF')
ax2.bar(Xlabel[14], data_plot[14], color = '#104E8B')
ax2.bar(Xlabel[15], data_plot[15], color = '#1E90FF')
ax2.bar(Xlabel[16], data_plot[16], color = '#104E8B')
ax2.bar(Xlabel[17], data_plot[17], color = '#1E90FF')

ax4.bar(Xlabel[18], data_plot[18], color = '#104E8B')
ax4.bar(Xlabel[19], data_plot[19], color = '#1E90FF')
ax4.bar(Xlabel[20], data_plot[20], color = '#104E8B')
ax4.bar(Xlabel[21], data_plot[21], color = '#1E90FF')
ax4.bar(Xlabel[22], data_plot[22], color = '#104E8B')
```

```
ax4.bar(Xlabel[23], data_plot[23], color = '#1E90FF')
ax4.bar(Xlabel[24], data_plot[24], color = '#104E8B')
ax4.bar(Xlabel[25], data_plot[25], color = '#1E90FF')

ax5.bar(Xlabel[26], data_plot[26], color = '#104E8B')
ax5.bar(Xlabel[27], data_plot[27], color = '#1E90FF')
ax5.bar(Xlabel[28], data_plot[28], color = '#104E8B')
ax5.bar(Xlabel[29], data_plot[29], color = '#1E90FF')
ax5.bar(Xlabel[30], data_plot[30], color = '#104E8B')

ax6.bar(Xlabel[31], data_plot[31], color = '#104E8B')
ax6.bar(Xlabel[32], data_plot[32], color = '#1E90FF')
ax6.bar(Xlabel[33], data_plot[33], color = '#104E8B')
ax6.bar(Xlabel[34], data_plot[34], color = '#1E90FF')
ax6.bar(Xlabel[35], data_plot[35], color = '#104E8B')
ax6.bar(Xlabel[36], data_plot[36], color = '#1E90FF')
ax6.bar(Xlabel[37], data_plot[37], color = '#104E8B')
ax6.bar(Xlabel[38], data_plot[38], color = '#1E90FF')

# if anonymous
#patient_number= 'xx'
#patient_ctrl_number= 'xx'

#voeg labels toe aan de grafiek en
ax1.set(xlabel='Visit '+patient_visit+' MDS-UPDRS', ylabel='MDS-UPDRS-score for subgroeps')
ax1.set_title("Patient "+patient_number+" MDS-UPDRS-scores during visit "+patient_visit,
fontweight = "bold", fontsize = 15)

ax2.set(ylabel='probability')
ax2.set_title("Patient "+patient_number+" smartwatch data during visit "+patient_visit,
fontweight = "bold", fontsize = 15)

ax3.set(xlabel='Visit '+patient_ctrl_visit+' MDS-UPDRS', ylabel='MDS-UPDRS-score for
subgroeps')
ax3.set_title("Patient "+patient_ctrl_number+" MDS-UPDRS-scores during visit
"+patient_ctrl_visit, fontweight = "bold", fontsize = 15)

ax4.set(ylabel='probability')
ax4.set_title("Patient "+patient_ctrl_number+" smartwatch data during visit
```

```
"+patient_ctrl_visit, fontweight = "bold", fontsize = 15)

ax5.set(xlabel='difference MDS-UPDRS', ylabel='MDS-UPDRS-score for subgroeps')
ax5.set_title("Difference MDS-UPDRS patient "+patient_ctrl_number+' visit '
+patient_ctrl_visit +' compared to patient '+patient_number + ' visit '+ patient_visit,
fontweight = "bold", fontsize = 15)

ax6.set(ylabel='probability')
ax6.set_title("Difference smartwatch data patient "+patient_ctrl_number+' visit '
+patient_ctrl_visit +' compared to patient '+patient_number + ' visit '+ patient_visit,
fontweight = "bold", fontsize = 15)

#plt.legend(fontsize = 8)
plt.show()


#%% statistical test between two visits of the same category
rvs1 = (c_trem_right,c_trem_left,c_brady_right,c_brady_left,c_gait_balance)
rvs2 = (v_trem_right,v_trem_left,v_brady_right,v_brady_left,v_gait_balance)
ttest_updrs = stats.ttest_rel(rvs1, rvs2,alternative='less') #considering it is expected that
the second visit 'comparinson' has a lower UPDRS due to the effectivness of the treatment

rvs3= (c_mean_tremor,c_mean_mild,c_mean_moderate,c_mean_none,c_mean_slight,
       c_mean_strong,c_mean_unknown,c_mean_dyskinesia)
rvs4= (v_mean_tremor,v_mean_mild,v_mean_moderate,v_mean_none,v_mean_slight,
       v_mean_strong,v_mean_unknown,v_mean_dyskinesia)
ttest_watch= stats.ttest_rel(rvs3, rvs4,alternative='less') #considering it is expected that
the second visit 'comparinson' has a lower UPDRS due to the effectivness of the treatment
#%% statistical test between two visits of different categories
temp1 = sum(rvs1)
rvs1mean=temp1/ len(rvs1)

temp2=sum(rvs2)
rvs2mean=temp2 / len(rvs2)

temp3=sum(rvs3)
rvs3mean=temp3 / len(rvs3)

temp4=sum(rvs4)
```

```
rvs4mean=temp4 / len(rvs4)

rvs5=(rvs1mean,rvs2mean)
rvs6=(rvs3mean,rvs4mean)

pearsonr_difference = stats.spearmanr(rvs5,rvs6)
```

8.2.14 09_graphics_10minutes

```
    # -*- coding: utf-8 -*-
"""
Created on Tue Jun 13 11:19:15 2023

@author: loet-
"""

# -*- coding: utf-8 -*-
"""
Created on Tue Jun  6 09:46:49 2023

@author: loet-
"""
import io
import os
import datetime as dt
from datetime import datetime as dtt
import pandas as pd
import matplotlib.pyplot as plt
from runeq import Config, stream
from scipy import stats
import numpy as np

BASE_PATH='G:/Datasets/overview/pt'
pt_number='10'
merged='_merged_all.xlsx'
df= pd.read_excel(BASE_PATH+pt_number+merged)
#%%
df=df.drop(["Unnamed: 0","Unnamed: 0.1"],axis=1)
```

```
#%%
df['time'] = pd.to_datetime(df['time'], unit='s')
#%%
hour = pd.to_datetime(df['time']).dt.hour
minute = pd.to_datetime(df['time']).dt.minute
#%%
time_window=input("select time_window(y/n):")
rest_time =' 00:00:00'
if time_window=='y':
    start_time = input("pattern; %d/%m/%Y example;29/11/2022:")
    start_time= dtt.strptime(start_time+rest_time,'%d/%m/%Y %H:%M:%S')
    end_time = input("pattern; %d/%m/%Y example;30/12/2022:")
    end_time= dtt.strptime(end_time+rest_time,'%d/%m/%Y %H:%M:%S')
    #end_time = input("pattern; %d.%m.%Y (example;'29.11.2022'):")
    #print (epoch)

    df = df[(df['time'] > start_time) & (df['time'] < end_time)]
elif time_window == "n":
    print('time_window equals whole period')
else:
    raise ValueError("choose one of the two options")
#%%
n = 23
dict={}
minutes = np.arange(0,61,10)
for i in range(0, n+1):
    for m in range(len(minutes)-1):
        time_mask = hour.eq(i) & minute.between(minutes[m],minutes[m+1]-1)
        dict[i,m]=df.loc[time_mask]

#%%
o=5
for i in range(0, n+1):
    for o in range (0, o+1):
        dict[i,o]=dict[i,o].drop(["mild","moder
        ate","strong","slight","none","unknown"],axis=1)
#%%
nn=143
availability = []
```

```
for i in range(0, nn+1):
    availability.append(i)
#%%
a=0
dik={}
for i in range(0, n+1):
    for o in range(0, o+1):
        dik[i,o]=dict[i,o].shape
        availability[a]=dik[i,o][0]
        a = a+1
#%%
b = 0
dun={}
for i in range(0, n+1):
    for o in range(0, o+1):
        dun[b]=dict[i,o]
        b = b+1

#%%
for i in range(0, nn+1):
    dun[i]=np.mean(dun[i])
#%%
len_plot = []
for i in range(0,nn+1):
    len_plot.append(i)
data_plot = len_plot
data_plot=[x/6 for x in data_plot]
#%%
my_list= []
for i in dun.values():
    my_list.append(i)
#%%
t = []
d = []
for i in range (0, nn+1):
    t.append(i)
    d.append(i)
    t[i]=my_list[i]["probability_tremor"]
    d[i]=my_list[i]["probability_dyskinesia"]
```

```
    # global total
    # total=t[i].__array__
    # print(total)
    #gh=t[i].append([])
    #%%
cumsum_vec =
np.cumsum(np.insert(availability, 0, 0))
window_width=6
ma_vec = (cumsum_vec[window_width:] -
cumsum_vec[:-window_width]) / window_width

xpoints=data_plot
xpoints2=data_plot[0:len(ma_vec)]
ypoints=t
y2points=d


fig, (ax1,ax2)=plt.subplots(2,1, figsize=[16,12])


#voeg labels toe aan de grafiek en
ax1.set(xlabel='Time:hour',
ylabel='Mean probability per 10 minutes')
ax1.set_title("Tremor & dyskinesia
probability for each 10 minutes",
fontweight = "bold", fontsize = 15)
ax1.set_xticks(np.arange(min(data_plot)
,max(data_plot),step=1.0))
ax1.plot(xpoints, ypoints)
ax1.plot(xpoints, y2points)
ax1.legend(['Tremor','Dyskinesia'],font
size = 15)


ax2.set_xticks(np.arange(min(data_plot)
,max(data_plot),step=1.0))
ax2.set(xlabel='Time:hour', ylabel='Data availability')
ax2.set_title("Data availability per
10 minutes", fontweight = "bold",
```

```
fontsize = 15)
# ax2.plot(xpoints2, ma_vec)
ax2.plot(xpoints, availability)
plt.show()
```