Rehabilitation Robotics and Machine Learning for Stroke Severity Classification

Raymond Greenfield

Follow this and additional works at: https://scholarworks.gsu.edu/math_theses

Recommended Citation
doi: https://doi.org/10.57709/32980067

This Thesis is brought to you for free and open access by the Department of Mathematics and Statistics at ScholarWorks @ Georgia State University. It has been accepted for inclusion in Mathematics Theses by an authorized administrator of ScholarWorks @ Georgia State University. For more information, please contact scholarworks@gsu.edu.
Rehabilitation Robotics and Machine Learning for Stroke Severity Classification

by

Raymond McCoy Greenfield

Under the Direction of Committee Chair’s Igor Belykh, Ph.D.

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in the College of Arts and Sciences

Georgia State University

2022
ABSTRACT

Stroke therapy is essential to reduce impairments and improve motor movements by engaging autogenous neuroplasticity. This study uses supervised learning methods to address an autonomous classification via stroke severity labeled data by a clinician. Thirty-three patients with chronic stroke performed a variety of rehabilitation activities while utilizing the Motus Nova rehabilitation technology to capture upper and lower body motion. Based on the minimum, maximum, and mean of the range of motion and pressure as well as the number of movements, force flexion, and extension for each game and session provided from the sensor data. Supervised learning methods were applied to a harmonized dataset of roughly 32,000 patient sessions based on the maximum score per session per game. With this approach using light gradient boosting methods we achieved an average of 94% accuracy with 10-fold cross-validation to prevent overfitting. This thesis shows objectively-measured rehabilitation training, enabling the identification of the stroke severity class with the hopes to have patients have a less severe class in the future.

Over the last 10 years robotic rehabilitation has been utilized in inpatient therapy. Robotic rehabilitation has been shown to be effective in improving the severity of stroke in some cases. In particular, robotic devices can be used to help stroke survivors regain movement, improve their functional abilities and improve depression (11). These devices can provide a high level of precision and repeatability, allowing patients to perform therapeutic exercises with greater accuracy and consistency (1). Additionally, because robotic devices can be programmed to provide different levels of assistance, they can be tailored to the individual needs of each patient. This allows for a more personalized and effective rehabilitation in-home program (21).

INDEX WORDS: Machine Learning, Artificial Intelligence, Deep Neural Network, Gradient Boosting, Physical Therapy, Neuroplasticity
Rehabilitation Robotics and Machine Learning for Stroke Severity Classification

by

Raymond McCoy Greenfield

Committee Chair: Igor Belykh
Committee: Vladimir Bondarenko, Alexandra Smirnova

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University

December 2022
DEDICATION

This thesis is dedicated to the stroke victims of the world in an effort to have a higher quality of life.
Above all, I offer my genuine gratitude to my major advisor Professor Igor Belykh and Dr. Rusell Jeter, both of who inspired me with their support and guidance. Without their advice and help, I could not have developed the knowledge and accomplished this thesis. Dr. Jeter has not only guided me into the statistical learning field but also taught me the way to learn new things which will allow me to continue my work and embrace new knowledge. I am also truly grateful to the members of my thesis committee: Professors Igor Belykh, Alexandra Smirnova, and Vladimir Bondarenko for their time and kind assistance during my thesis preparation.

I would also like to thank Dr. Xiaojing Ye and Dr. Sergey Plis for their courses on optimization and deep learning respectively. I genuinely learned a lot from them in their courses and were honestly some of my favorites. Both pushed me to not only understand the rigorous mathematics that backs the frameworks that are used in modern machine-learning methods. My heartiest appreciation also goes to all the faculties and staff of the department of Mathematics Statistics who contributed in diverse ways to make my experience a token of memory at Georgia State University.

Lastly, I would like to express my thanks to my mother Julie Greenfield RPH for her many manifold blessings and support to me. Without them, I would not have been able to complete the pursuit of my thesis studies.

This work was supported by the NSF under Grant CMMI-1953135.
# TABLE OF CONTENTS

ACKNOWLEDGMENTS ......................................................... v

LIST OF TABLES ............................................................... viii

LIST OF FIGURES ............................................................. ix

1 Introduction ................................................................. 1
  1.1 Background .......................................................... 1
  1.2 Methods and Procedure ............................................. 5

2 Data Collection and Harmonization ..................................... 8
  2.1 Data Collection ..................................................... 8
  2.2 Clinician Labeling .................................................. 8
    2.2.1 Clinician Label Mapping ................................... 9
  2.3 Data Harmonization ................................................ 11

3 Exploratory Data Analysis .............................................. 13

4 Methodologies ............................................................. 15

5 Computational Experiments ............................................. 18
  5.1 Model Description .................................................. 18
    5.1.1 Light Gradient Boosting .................................... 18
    5.1.2 Extra Trees Classifier ....................................... 20
    5.1.3 Multi-Layer Perceptron (Deep Neural Network) .......... 21
    5.1.4 Logistic Regression ........................................... 21
  5.2 Results ............................................................. 22
    5.2.1 Determining Relevant Features .............................. 24

6 Conclusion ................................................................. 27
Appendices ................................................................. 29

A  Data Dictionary ...................................................... 30

B  Python Code .......................................................... 31
   A  TallDataFrame.py .................................................. 31
   B  max_score_per_session.py ........................................ 36

REFERENCES .............................................................. 47
LIST OF TABLES

Table 1.1  Example patient label table assessed by a clinician using a potentiometer (12). Note that the final label is at the discretion of the clinician and could be based on qualitative factors not accounted for in the test. . . . . . . . . . 7

Table 5.1  10-Fold Cross-Validation Mean Scores with Standard Deviation . . . . 24

Table 1  Data Dictionary Table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
LIST OF FIGURES

Figure 1.1 Patients do therapy sessions with the Motus Hand or Motus Foot using a pneumatically-driven exogenous robotic device that is worn on the affected hand/arm (depicted in the bottom panel). The peripheral acts as a game controller (through an angle sensor embedded in the wrist joint) that allows users to play therapeutic video games that are able to dynamically adapt to their needs and provide them the requisite assistance/resistance (computer screen in the bottom panel).

Figure 1.2 Concept diagram of the overall data science process. Data is gathered from therapy sessions performed using the Motus Hand and Motus Foot. The data is processed and used in a supervised machine learning model to classify the stroke severity of the patient. A total of 11 predictor variables attribute to the classification of stroke severity.

Figure 2.1 Mapping of Clinician Label based on Min, Max, Peripheral Type, and Assessment Variables.

Figure 3.1 The minimum pressure variable is not displayed since the majority of the values are zero. This can be seen in the average pressure and maximum pressure variables.

Figure 3.2 Value counts of the peripheral types and classes.

Figure 4.1 The correlation matrix does not include key variables such as Patient ID, Session ID, Game ID, and Start Time. Importance in dimensionality reduction based on a greater than 0.9 threshold. As seen above, $F_{flex}$ and $F_{ext}$ are highly negatively correlated. However, these variables were both used in the analysis.

Figure 4.2 Depiction of the Principal Components with the Explained Variance Ratio. As shown 95% of the explained variance is contributed by all principal components (8). As a result, all variables are used in the machine learning model for the analysis.

Figure 5.1 LGB classification stump. There are 139 stumps in the model. Due to the complexity of the tree the labels are difficult to read.
Figure 5.2 Several different architectures of the DNN can be applied. In this case, the dimensionality reduction techniques implied keeping the same number of dimensions for the hidden layers. Here, there are 3 hidden layers with dimensions (8,5,8) respectively.

Figure 5.3 Accuracy Score for each Machine Learning Model. As shown, the Light Gradient Boosting (LGB) and Extra Trees have the greatest accuracy score. See Table 5.1 for the depiction of the cross-validation to prevent over-fitting (2). Note that accuracy alone is not the best illustration for the performance of a machine learning model. LGB is chosen as the final model in extension for future work.

Figure 5.4 Confusion Matrices Light Gradient Boosting and Logistic Regression. Considering the false negative column of the No classification, it is seen that the LGB model greatly improved this classification. This is especially important when classifying a patient as having No stroke severity when they are actually a High severity.

Figure 5.5 Classification Report Extra Trees, Light Gradient Boosting, Logistic Regression, Neural Network

Figure 5.6 Feature Importances of each model displaying which variables make the greatest contributions to the model. As shown the $F_{ext}$ and $F_{flex}$ make the most contributions to the model. Here we also see that the peripheral type (Hand or Foot) made the least. This could be due to the fact that only two patients used the foot as a lower extremity.
1.1 Background

Stroke is a leading cause of mortality and disability worldwide and the economic costs of treatment and post-stroke care are substantial (5). In 2019, there were 12.2 million incident cases of stroke, 101 million prevalent cases of stroke, 143 million (DALYs) due to stroke, and 6.55 million deaths from stroke (6). The severity of a stroke can range from mild to severe, with severe strokes often leading to long-term disability or even death. Stroke rehabilitation typically involves a team of healthcare professionals, including doctors, nurses, therapists, and other specialists. The specific goals and interventions of stroke rehabilitation will vary depending on the individual’s needs and abilities, but may include physical therapy to improve mobility, occupational therapy to improve the ability to perform daily activities, speech therapy to improve communication skills, and cognitive therapy to improve memory, problem-solving, and other cognitive abilities. Stroke rehabilitation typically involves a team of healthcare professionals, including doctors, nurses, therapists, and other specialists. The specific goals and interventions of stroke rehabilitation will vary depending on the individual’s needs and abilities but may include physical therapy to improve mobility, occupational therapy to improve the ability to perform daily activities, speech therapy to improve communication skills, and cognitive therapy to improve memory, problem-solving, and other cognitive abilities. Our goal is to transfer this now not only to an in-home setting for the patient but use machine learning to quantify the progress of patient improvement.
Supervised learning is a type of machine learning in which a model is trained on a labeled dataset, where the correct output (or label) is provided for each input. The goal of supervised learning is to build a model that can make accurate predictions on unseen data.

To use supervised learning methods to classify stroke severity, you would need to collect a dataset of stroke patients, where each patient is labeled with their stroke severity (e.g. mild, moderate, severe). This dataset would then be used to train a machine learning model.

There are many different machine learning algorithms that could be used for this task, including decision trees, support vector machines, and neural networks. The choice of algorithm would depend on the specific characteristics of the dataset, such as the number of samples, the number of features, and the distribution of the classes.

Once the model is trained, it can be used to make predictions on new, unseen data. For example, given information about a patient’s symptoms and medical history, the model
could predict the severity of their stroke. These predictions could then be used to guide treatment and prognosis.

The Motus Hand and Foot can be used without needing help from a clinician or caregiver. Motus technology uses high-dose repetitive task practice to induce neuroplasticity to help stroke survivors improve neuroplasticity in the upper and lower peripherals (13).

Patients take part in video games during a therapy session and would receive a score based on their performance in the game. Using artificial intelligence the Motus Nova robotic Hand or Foot applies pressure to the hand or foot to increase the range of motion during the session (22).

The recent growing interest is due to the increasing complexity and numerosity of available data sets, for which more classical methods do not allow accurate results as shown in this study (3),(18). Here, we provide a unique dataset collected from patient therapy sessions. Machine Learning is the methodology that provides computers with the ability to learn from experience i.e. training a model on a split dataset (3).

Utilizing ML and AI may have a central role in rehabilitation decision-making in determining if patient therapy is improving. ML is the methodology that provides computers with the ability to learn from experience. By constructing and training supervised classifiers to learn decision rules from data, automatic solutions able to make predictions on new data can be exploited.

Like in many healthcare, disease, or machine learning research applied in a clinical setting, labeling of patient data by a clinician is necessary (16). This study applies the same heuristic
methodologies.

Figure 1.2 provides a high-level overview of the data collection, analysis, processing, and modeling that produce the final classification results.

Figure 1.2: Concept diagram of the overall data science process. Data is gathered from therapy sessions performed using the Motus Hand and Motus Foot. The data is processed and used in a supervised machine learning model to classify the stroke severity of the patient. A total of 11 predictor variables attribute to the classification of stroke severity.
1.2 Methods and Procedure

The Motus Hand and the Motus Foot each consist of two major components: a peripheral that the patient attaches to their affected limb, and an interactive console that guides their therapy routine and assessment using a video game interface. The peripherals are equipped with a pneumatic actuator that is able to dynamically provide assistance/resistance by filling an air muscle located in the peripheral that then moves the wrist/ankle joint of the peripheral. The wrist/ankle joint of the peripheral has an embedded angle and pressure sensor that transmits live angle data to the console. This allows the console to give the user immediate visual feedback of their movement through avatars in a video game on the screen. The therapeutic video game activities are able to provide a dynamic feedback loop consisting of in-game goals (ships to shoot, for example) that drive user movements, which correspond to movement on-screen which allow the console to react and set new goals/obstacles. This feedback loop is designed to promote sensory-motor function.

A therapy session with the Motus Hand or Foot consists of a variety of stretching, gross motor control, fine motor control, and endurance exercises depending on the patient’s needs. This process is depicted in Figure 1.1 where a Motus Hand user is playing "Cosmic Tennis," which is a gross motor control exercise that plays like the classic arcade game Pong 1.1. The user’s wrist/ankle movement corresponds to the movement of the paddle on the right-hand side of the screen, and the goal is to hit the ball back and forth to score on the AI-controlled opponent.

In order to use the data collected during a therapy session to autonomously classify a
patient’s stroke severity, each patient was given a guided assessment with a clinician using the Motus Hand or Foot to classify them as having High Range of Motion (ROM), Low ROM, or No ROM.

Using four machine learning algorithms: light gradient boosting, extra trees classifier, deep neural network (DNN), and logistic regression. A practical model using the most common data measured in each session data based on the maximum score per session per patient. Unsupervised learning methods were applied to the final dataset such as the correlation matrix and principal component analysis to show that all variables collected are relevant to the study. This includes a 10-fold cross-validation on the final dataset of size 16 columns and 32,902 rows (sessions) with the mean and standard deviation of accuracy from each computational experiment. From here the following metrics are used to determine the performance of the model including the accuracy, precision, and recall from the confusion matrix. The macro average f1-score was accounted for this multi-classification problem (4).

In summary, session data is collected, processed, and analyzed sequentially. Based on the performance measures mentioned before the light gradient boosting model best fit this dataset with more than a 50% improvement compared to other classical methods such as logistic regression.
<table>
<thead>
<tr>
<th>ID</th>
<th>Max</th>
<th>Min</th>
<th>Assessment</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>2054</td>
<td>30</td>
<td>3</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>1495</td>
<td>37</td>
<td>-20</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>2058</td>
<td>50</td>
<td>-24</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>2273</td>
<td>21</td>
<td>-16</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>2085</td>
<td>40</td>
<td>-15</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>2098</td>
<td>44</td>
<td>-9</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>1864</td>
<td>28</td>
<td>-12</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>1859</td>
<td>45</td>
<td>-17</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>1479</td>
<td>45</td>
<td>-20</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>1865</td>
<td>30</td>
<td>-15</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>2128</td>
<td>34</td>
<td>-10</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>2138</td>
<td>33</td>
<td>-15</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>2183</td>
<td>41</td>
<td>-16</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>2040</td>
<td>37</td>
<td>-18</td>
<td>Passive</td>
<td>No</td>
</tr>
<tr>
<td>2097</td>
<td>43</td>
<td>-18</td>
<td>Passive</td>
<td>Low</td>
</tr>
<tr>
<td>2356</td>
<td>-3</td>
<td>-17</td>
<td>Assisted</td>
<td>Low</td>
</tr>
<tr>
<td>2356</td>
<td>-3</td>
<td>-17</td>
<td>Assisted</td>
<td>Low</td>
</tr>
<tr>
<td>1688</td>
<td>52</td>
<td>-23</td>
<td>Assisted</td>
<td>Low</td>
</tr>
<tr>
<td>1876</td>
<td>54</td>
<td>-12</td>
<td>Passive</td>
<td>Low</td>
</tr>
<tr>
<td>2029</td>
<td>46</td>
<td>-20</td>
<td>Passive</td>
<td>Low</td>
</tr>
<tr>
<td>1458</td>
<td>30</td>
<td>-18</td>
<td>Passive</td>
<td>Low</td>
</tr>
<tr>
<td>1113</td>
<td>33</td>
<td>-20</td>
<td>Assisted</td>
<td>High</td>
</tr>
<tr>
<td>2282</td>
<td>38</td>
<td>-13</td>
<td>Assisted</td>
<td>High</td>
</tr>
<tr>
<td>1057</td>
<td>10</td>
<td>-12</td>
<td>Assisted</td>
<td>High</td>
</tr>
<tr>
<td>2282</td>
<td>8</td>
<td>-16</td>
<td>Assisted</td>
<td>High</td>
</tr>
<tr>
<td>1781</td>
<td>39</td>
<td>-15</td>
<td>Assisted</td>
<td>High</td>
</tr>
<tr>
<td>2360</td>
<td>10</td>
<td>-18</td>
<td>Assisted</td>
<td>High</td>
</tr>
<tr>
<td>2035</td>
<td>41</td>
<td>7</td>
<td>Assisted</td>
<td>High</td>
</tr>
<tr>
<td>1799</td>
<td>48</td>
<td>1</td>
<td>Assisted</td>
<td>High</td>
</tr>
<tr>
<td>2191</td>
<td>40</td>
<td>-20</td>
<td>Assisted</td>
<td>High</td>
</tr>
<tr>
<td>1974</td>
<td>38</td>
<td>-6</td>
<td>Assisted</td>
<td>High</td>
</tr>
<tr>
<td>2004</td>
<td>41</td>
<td>-20</td>
<td>Assisted</td>
<td>High</td>
</tr>
<tr>
<td>2179</td>
<td>49</td>
<td>12</td>
<td>Assisted</td>
<td>High</td>
</tr>
<tr>
<td>1470</td>
<td>20</td>
<td>-16</td>
<td>Assisted</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 1.1: Example patient label table assessed by a clinician using a potentiometer (12). Note that the final label is at the discretion of the clinician and could be based on qualitative factors not accounted for in the test.
CHAPTER 2
Data Collection and Harmonization

2.1 Data Collection

Throughout a therapy session using the Motus Hand or Motus Foot, live angle data (measured in degrees from a natural midpoint in wrist/ankle placement) is collected from the sensor embedded in the wrist or ankle joint at 50 readings per second. This “raw” angle sensor data is then stored in a high resolution time series database (InfluxDB (15)). In addition to the high resolution angle data, pressure readings (measured in PSI) are taken from the pressure management system at 50 measurements per second.

Each therapy session a patient does includes a selection of $\approx 30$ activities that focus on different types of motor function including: gross motor control, fine motor control, flexor tone reduction, endurance, reaction time, and tracking.

A patient can participate in more than one video game during a patient session. Once the patient is finished with the video game the score is recorded and stored.

Gender and other biometric data such as age, height and weight are not included in the patient description or the analysis.

2.2 Clinician Labeling

In order to have a supervised learning method for the machine learning algorithm, there has to be human labeling. Quantitative and qualitative factors at the discretion of the clinician go into the labeling of stroke patients. By using the potentiometer (12), clinicians are able
to gather data on the patient in Table 1.1. We define a patient as Assisted if the clinician themselves applies pressure to the patient to achieve the minimum and maximum angle. Similarly, patients are Passive meaning that the clinician does not help them achieve this maximum or minimum angle in the labeling process. The low label is mixed between the assisted and passive. Notice that all patients that are classified with a high stroke severity were assisted by the clinician. This is important when noticing that patients 2085 and 1781 (Blue) have a similar total range of motion (Min + Max), but patient 1781 is assisted by the clinician. However, there is ambiguity in some labels. For example, take patient 2356 (red) where it can be argued that the patient should have a high stroke severity given the low total range of motion with the assistance. This is where the clinician has other outside factors that contribute to the final labeled classification of a patient.

2.2.1 Clinician Label Mapping

In efforts to quantify the labels that the clinician provided to the patients. A mapping based on the minimum and maximum angle, the range of motion assessment (Active or Passive), and the peripheral type were used as variables in the analysis. The assessment is done by using a potentiometer (12) This is simply to check if there is consistency in the clinician labeling and for later purposes, as more and more patient labels are collected it can possible to classify more patients without necessarily having the clinician label. In this case, there are 50 patients. The 33 patients are users with the score data. Unfortunately, a strong classification was not achieved with a performance of less than 80% as shown in Figure 2.1. In the future, more patient data is gathered this could become more prominent.
Figure 2.1: Mapping of Clinician Label based on Min, Max, Peripheral Type, and Assessment Variables

Deep learning can be more of a blackbox approach, especially with only 50 data-points. The following is an example of the architecture using in testing for clinician label mapping.
2.3 Data Harmonization

Each game had an original column and associate min, max, and mean range of motion and pressure causing the dataset to be rather wide (72 columns) and sparse since each patient cannot necessarily play every game in every rehab session. In order to combat this, we must perform a data transformation. For example, we have the following six variables for Plinko;

\[
\begin{align*}
Plinko_{\text{rom\_min}} & \rightarrow game_{\text{rom\_min}}, \\
Plinko_{\text{rom\_max}} & \rightarrow game_{\text{rom\_max}}, \\
Plinko_{\text{rom\_mean}} & \rightarrow game_{\text{rom\_mean}}, \\
Plinko_{\text{pressure\_min}} & \rightarrow game_{\text{pressure\_min}}, \\
Plinko_{\text{pressure\_max}} & \rightarrow game_{\text{pressure\_max}}, \\
Plinko_{\text{pressure\_mean}} & \rightarrow game_{\text{pressure\_mean}} 
\end{align*}
\]

This is then mapped for all thirty games with the associated game_id column with a key value of 1-30. Each game variable was mapped to a row and the dataset then had a general range of motion and pressure variable corresponding to each patient, session, and game. Hence, we go from having a sparse wide dataset to a tall dataset with missing values, but not a sparse matrix.

There were five main cases of missing data or data misrepresentation. The first three cases included the $R_{\text{min}}, R_{\text{max}},$ and the $t_{\text{game}}$ (game time) variables with missing values. Since we could not necessarily deal with taking averages on the patient level we considered the mean of the column and filled the values accordingly. the game (game time) variable we took the
minimum value. Another assumption was the $N_{mov}$ variable (number of movements in a session) where we placed a floor at the minimum number of movements at 3. A movement is considered a full flexion or extension of the hand or foot in a patient session. This is mainly under the assumption that the patient did not actually commit to a therapy session without a certain number of movements. In the last case for some patients, the sensor read a negative value for the minimum pressure value. This, of course, is not physically possible and in these cases, the rows with negative pressure were clipped and set to zero meaning no pressure was applied by the robotic hand or foot during the rehab session.

Finally, aggregating the dataset based on the maximum score per game per session generated a dataset of size 32,902 rows with 16 columns.

For the final dataset the maximum score for each therapy session per game for each patient. In this dataset, we have the following description of the columns: the game patient, and session identification with the start time associated with these. The minimum, maximum, and mean pressure (PSI) and range of motion along with the amount of time the game is played and the score of the game. Each of these patients has an associated peripheral and classification. In the summary statistics of the final dataset, there were a few missing points that needed to be addressed. This includes the matter that there were negative pressure readings from the sensor data. Since this is not actually possible the data was floored and assumed that the negative pressures were reading at zero pressure. The range of motion measurement has negative values. This is from the sensor when gathering data where the range of the angle is $-25^\circ$ to $55^\circ$. 
CHAPTER 3  
Exploratory Data Analysis

The next step for any data science project after collection and harmonization is to explore the data and gather insights. One of the properties to distinguish the distribution of the data is shown in Figure 3.1. These variables have to be normally distributed in order to use a machine learning algorithm because of scaling (10).

Figure 3.1: The minimum pressure variable is not displayed since the majority of the values are zero. This can be seen in the average pressure and maximum pressure variables.

It is noted that the distribution of a random variable needs to be normally distributed to follow the central limit theorem in order to use a machine learning algorithm. Though one can argue the lack of normal distribution of the force flexion and extension variable. Later
it is shown to have little effect on the final results of the classification analysis (see chapter 5).

The following Figure 3.2 shows a count of the peripheral types and the number of patients in each class (High, Low, No). As seen many of the patients used the Motus Hand and only 3 used the Foot. In future studies, there will be more usage of the foot. Also, note that there are more patients in the No ROM class. This is at the discretion of the clinician during a patient test with a potentiometer. The patient with the No ROM class all received no assistance during the test, while on the other hand patients with the High ROM class received assistance from the clinician. The bar charts are from table 1.1.

![Figure 3.2: Value counts of the peripheral types and classes.](image-url)
CHAPTER 4
Methodologies

Unsupervised Learning Methods: Correlation Matrix and Principal Component Analysis

One of the goals of this study is to determine if there exists a key indication variable that is a direct indicator of stroke classification. Here we use this unsupervised learning allows us to see if there is a need for a dimensionality reduction in the analysis (8).

The correlation matrix in Figure 4.1 is an indication of dimensionality reduction inherently. We disregard key variables such as Patient ID, Game ID, and Start Time since those are indications of the start of a patient session. If two variables are highly correlated i.e. $\text{Cor}(X, Y) > 0.9$, then this is a good indication that one of those variables can be dropped in our analysis (19). In Figure 4.1, we can consider the lower or upper half of the matrix. There exists a strong negative correlation between $F_{ext}$ and $F_{flex}$, but intuitively it does not seem useful to drop either one of the variables in the analysis individually. The correlation between the game pressure mean and max with the value of 0.80, is an indication that more pressure is being applied by the Motus Hand or Foot on average in each session. Since this correlation does not surpass the threshold of 0.90 we do not drop either variable. In consideration of the case where the $t_{game}$ (game time) variable has a correlation of 0.60, this is an indication that the longer a patient plays a game i.e. the longer the session time this should increase the score received.

Consider Figure 4.2, here we graph each principal component and how much variation that eigenvector contributes to the total variation. Principal components are new variables
that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components. Explained variance is a statistical measure of how much variation in a dataset can be attributed to each of the principal components (eigenvectors) generated by the Principal Component Analysis (PCA) method. It tells us how much of the total variance is “explained” by each component. This allows us to rank the components in order of importance, and to focus on the most important ones when interpreting the results of our analysis (9).

From here we can interpret the following, that all 11 variables are needed for the analysis. Hence, we cannot fit a better model (see final results) that has a high accuracy and performance measure.

![Correlation Matrix](image)

Figure 4.1: The correlation matrix does not include key variables such as Patient ID, Session ID, Game ID, and Start Time. Importance in dimensionality reduction based on a greater than 0.9 threshold. As seen above, $F_{\text{flex}}$ and $F_{\text{ext}}$ are highly negatively correlated. However, these variables were both used in the analysis.
Figure 4.2: Depiction of the Principal Components with the Explained Variance Ratio. As shown 95% of the explained variance is contributed by all principal components (8). As a result, all variables are used in the machine learning model for the analysis.
CHAPTER 5
Computational Experiments

5.1 Model Description

Gradient boosting decision tree (GBDT) is a widely-used machine learning algorithm, due to its efficiency, accuracy, and interpretability (7). In essence, the algorithm uses smaller "weaker classifiers" with a number of leaves. Taking a weighted average of these several "weaker classifiers" we are able to construct a "stronger classifier". This is by the weak learners theorem (17). This will be the key model used in our analysis. From the lightgbm module in python version 3.9.10 we used the LGBMClassifier as well as for the other models. The next four sections will be on the model descriptions used in the python code.

For example, the n-estimators parameter is the number of "weaker" models used in the construction of the model. The construction of the model came from Microsoft GitHub with auto machine learning libraries (14). Hence, the parameters chosen above came from the algorithm in the library FLAML. FLAML stands for "Flexible Large-scale Automated Machine Learning." It is a machine learning library developed by Microsoft that allows users to easily train and deploy machine learning models on large datasets.

5.1.1 Light Gradient Boosting

Below is an general approach to how the algorithm is applied.

Input: training set \{ (x_i, y_i) \}_{i=1}^n, a differentiable loss function \( L(y, F(x)) \), number of
iterations $M$. Algorithm: 1. Initialize the model with a constant value:

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For $m = 1$ to $M$:
   1. Compute so-called pseudo-residuals: for $i=1, \ldots, n$.

   $$r_{im} = -\left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)}.$$

   2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.

   3. Compute multiplier $\gamma_m$ by solving the following one-dimensional optimization problem:

   $$\gamma_m = \arg \min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)).$$

   4. Update the model:

   $$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

```python
from lightgbm import LGBMClassifier
LGBMClassifier(coldsample_bytree=0.5332477358865868,
               learning_rate=0.4072454511649998,
```
max_bin=255, min_child_samples=2, n_estimators=139, num_leaves=115, reg_alpha=0.006958608037974516, reg_lambda=0.0009765625, verbose=-1)

Figure 5.1: LGB classification stump. There are 139 stumps in the model. Due to the complexity of the tree the labels are difficult to read.

5.1.2 Extra Trees Classifier

Extra Trees Classifier:

from sklearn.ensemble import ExtraTreesClassifier

ExtraTreesClassifier(criterion='entropy', max_features=1.0, max_leaf_nodes=8717, n_estimators=42, n_jobs=-1
5.1.3 Multi-Layer Perceptron (Deep Neural Network)

```python
from sklearn.neural_network import MLPClassifier

MLPClassifier(activation = 'relu', alpha = 0.005,
               hidden_layer_sizes = (8,5,8), learning_rate = 'constant',
               solver = 'adam', max_iter=1000)
```

Figure 5.2: Several different architectures of the DNN can be applied. In this case, the dimensionality reduction techniques implied keeping the same number of dimensions for the hidden layers. Here, there are 3 hidden layers with dimensions (8,5,8) respectively.

5.1.4 Logistic Regression

The default architecture was used from the documentation website.
5.2 Results

The goal of our computational experiments was to compare the performance of different machine learning methods for the identification of stroke severity. The original harmonized dataset contained all the scores, minimum and maximum range of motion, and minimum and maximum pressure and we took the maximum score per game per session. Because of the smaller-sized dataset, the training and testing were split on the 80/20 principle where 20% of the data was the testing data (20).

Train Shape: (26321, 11)

Test Shape: (6581, 11)

In Table 5.1, we see a 10-fold cross-validation of each of the machine learning classification algorithms with Figure 6 as a visual display of a single accuracy measure. This was done on the chance that our model of high accuracy is over-fitting our model. This is where the dataset is randomly divided into 10 different subsets with 320 rows each (2). Each ”folded” dataset becomes the new training data another becomes the testing data. The same model is used to fit the dataset and then tested on the random testing dataset. We then take the mean and standard deviation of the random sample of the accuracy.

In Figure 5.4, we see the confusion matrix of each of the supervised learning methods. Generally, a confusion matrix is used to visually represent the performance of the algorithm. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa. We represent the percentage over
the actual numeric number for display purposes. Three metrics of performance come from the confusion matrix precision, recall, and the f1-score. Precision measures the proportion of predicted positives that are truly positive. Recall measures the proportion of predicted negatives that are truly negative. The f1-score is the harmonic mean of the precision and recall (4). In this case, this is macro-averaging (treating all classes as equally important). The interesting portion here is that the accuracy of the Extra Trees classifier is higher than the Light Gradient Boosting Method (LGBM), but when comparing the f1-score the LGBM is (in this case) the better classifier.

![Classifier Accuracy Scores](image)

Figure 5.3: Accuracy Score for each Machine Learning Model. As shown, the Light Gradient Boosting (LGB) and Extra Trees have the greatest accuracy score. See Table 5.1 for the depiction of the cross-validation to prevent over-fitting (2). Note that accuracy alone is not the best illustration for the performance of a machine learning model. LGB is chosen as the final model in extension for future work.
Table 5.1: 10-Fold Cross-Validation Mean Scores with Standard Deviation

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Mean</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extra Trees</td>
<td>96.40%</td>
<td>0.4%</td>
</tr>
<tr>
<td>Light Gradient Boosting</td>
<td>94.0%</td>
<td>0.4%</td>
</tr>
<tr>
<td>Neural Network</td>
<td>71.70%</td>
<td>0.7%</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>61.20%</td>
<td>0.5%</td>
</tr>
</tbody>
</table>

In order to prevent over-fitting 10-fold cross-validation is performed on the dataset (2). This is done by slicing the data into 10 different training and testing datasets and then training the model on each of those 10 ”sliced” datasets. The average score is taken from each of the models to indicate performance.

In the classification report, the f1-score for light gradient boosting receives an accuracy of 96%. This is an indication this is the model of choice. When considering the confusing matrix, the model does a better job of not classifying patients as having No ROM when the patient actually has a High ROM. When a model is wrong there needs to be a way to minimize this. Light gradient boosting does the best out of all the models along with high performance.

5.2.1 Determining Relevant Features

When using a bagging algorithm, the final step is to analyze the features of the model contributing most to the classification. There are two of these methods used: Extra Trees and Light Gradient Boosting. In Figure 5.6, notice that both algorithms have similar results.
Figure 5.4: Confusion Matrices Light Gradient Boosting and Logistic Regression. Considering the false negative column of the No classification, it is seen that the LGB model greatly improved this classification. This is especially important when classifying a patient as having No stroke severity when they are actually a High severity.

Interestingly enough, the score variable contributed little in comparison to the other variables. This was against most intuition. The peripheral type as well contributed the least, which is most likely because most of the patients used the Motus hand in the patient sessions. This is speculation. On the other hand, the number of movements, force extension and flexion have the most contributions to the model. This is very important since this
Figure 5.5: Classification Report Extra Trees, Light Gradient Boosting, Logistic Regression, Neural Network

<table>
<thead>
<tr>
<th></th>
<th>Low</th>
<th>High</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>accuracy</td>
<td>96.70%</td>
<td>96.70%</td>
<td>96.70%</td>
</tr>
<tr>
<td>macro avg</td>
<td>96.70%</td>
<td>96.70%</td>
<td>96.70%</td>
</tr>
<tr>
<td>weighted avg</td>
<td>96.70%</td>
<td>96.70%</td>
<td>96.70%</td>
</tr>
</tbody>
</table>

Figure 5.6: Feature Importances of each model displaying which variables make the greatest contributions to the model. As shown the $F_{(ext)}$ and $F_{(flex)}$ make the most contributions to the mode. Here we also see that the peripheral type (Hand or Foot) made the least. This could be due to the fact that only two patients used the foot as a lower extremity.

implies that the more the patients move their hand or foot, the greater contribution to their stroke classification! Although this does not imply (in this case) moving their hand decreases the stroke severity level. A time series analysis is needed.
CHAPTER 6

Conclusion

This study uses 4 computational methods via supervised learning methods to classify stroke severity for patients. From this, there are not standard methods (logistic regression), black box methods (Deep Neural Network), and bagging methods (light gradient boosting and extra trees). There are many benefits to the bagging methods, one of them being interpretability. This is different from using a neural network, which may or may not be a stronger algorithm in the future. The interpretability of a model (see feature importances 5.6) allows there to be an interpretation of which variables contribute to the classification. This is assuming high accuracy and lack of over-fitting (seeing cross-validation table). For example, as stated before, the score variable has a small contribution although game time is highly contributed (both variables have a correlation of 0.60, interesting.

Robotic in-home patient therapy allows stroke patients to battle their illnesses in a more effective way. Not only do patients lack the ability to do day-to-day tasks like eating or tying a shoe, but they can also simply struggle to see a clinician without a caregiver with the inability to operate a motor vehicle. This can be a financial toll on the patient as well. With the combination of in-home robotic therapy and machine learning, there could be potential to do this in an autonomous way.

Predictive models can be a promising support tool for clinicians. Machine Learning algorithms can be easily deployed for this purpose, due to their capability of handling large cohorts and high dimensional datasets. Once trained, they provide accurate estimates at a
low cost. Among these advantages, this kind of solution could stimulate a more data-driven approach in clinical practice, promote a more structured definition of studies and reduce the gap between clinical and research areas. For this reason, more data will be gathered in time, with more analyzes used such as time-series approaches to lead to further embedded decision-support tools for the daily use of not only the clinician but the patient as well.

In another study, there can be a greater quantitative understanding of patient therapy improved through robotic therapy. This can be done by simply having more session data gathered. Potentially taking a group of patients and monitoring the timer series analysis to see when a stroke severity changes from High to Low or Low to No. There could even be a web application for each individual patient to track progress. This could allow more data to be gathered faster, but patient labeling would have to be done separately.

This could allow patients to track their progress in real time on a weekly and monthly basis. Insights with these models can offer a practical guideline for future design for improved treatment. As more and more session data is gathered there can even be a single diagnosis of which a patient should be improving to have more effective treatment on the individual level.
Appendices
## A Data Dictionary

Table 1: Data Dictionary Table

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Unit</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Time</td>
<td>Start time of the session</td>
<td>Datetime</td>
<td>31:44.2</td>
</tr>
<tr>
<td>Game ID</td>
<td>Unique game identifier assigned the 30 games</td>
<td>0-29</td>
<td>16</td>
</tr>
<tr>
<td>Session ID</td>
<td>Unique identifier for a therapy session</td>
<td>RNG</td>
<td>338057214367</td>
</tr>
<tr>
<td>Patient ID</td>
<td>Unique user identifier preserved across Motus Nova databases</td>
<td>integer</td>
<td>1113</td>
</tr>
<tr>
<td>$F_{\text{flex}}$</td>
<td>Maximum centripetal force generated moving in the downward direction (computed from derivatives of angle data)</td>
<td>neutons</td>
<td>-3.047709105</td>
</tr>
<tr>
<td>$F_{\text{ext}}$</td>
<td>Maximum centripetal force generated moving in the upward direction (computed from derivatives of angle data)</td>
<td>neutons</td>
<td>3.251405759</td>
</tr>
<tr>
<td>$N_{\text{mov}}$</td>
<td>The number of completed movements</td>
<td>integer</td>
<td>10</td>
</tr>
<tr>
<td>$R_{\text{min}}$</td>
<td>Absolute minimum angle detected by angle sensor during therapy session</td>
<td>degrees</td>
<td>-25</td>
</tr>
<tr>
<td>$R_{\text{max}}$</td>
<td>Absolute maximum angle detected by angle sensor during therapy session</td>
<td>degrees</td>
<td>46.41941</td>
</tr>
<tr>
<td>$t_{\text{game}}$</td>
<td>Total time spent performing therapy during a session for that game</td>
<td>seconds</td>
<td>15</td>
</tr>
<tr>
<td>$P_{\text{min}}$</td>
<td>Minimum Pressure applied by the sensor in a patient session</td>
<td>psi</td>
<td>-0.04511994</td>
</tr>
<tr>
<td>$P_{\text{max}}$</td>
<td>Maximum pressure applied by the sensor in a patient session</td>
<td>psi</td>
<td>10.30989</td>
</tr>
<tr>
<td>$P_{\text{mean}}$</td>
<td>Average pressure applied by the sensor in a patient session</td>
<td>psi</td>
<td>3.590553432</td>
</tr>
<tr>
<td>Score</td>
<td>Score achieve by patient per video game</td>
<td>integer</td>
<td>100</td>
</tr>
<tr>
<td>$h$</td>
<td>Peripheral type variable indicating the Hand or Foot</td>
<td>0, 1</td>
<td>Hand</td>
</tr>
<tr>
<td>Class</td>
<td>Designate stroke severity label for a patient by a clinician (High, Low, No)</td>
<td>0, 1, 2</td>
<td>High</td>
</tr>
</tbody>
</table>
B Python Code

Displaying all of the code used in the analysis would take over 50 pages of this thesis. Here is a link to the Github repository: https://github.com/rgreenfield/GSU_MotusNova_Data

A TallDataFrame.py

```python
def tall_df(data, game_dict):

    #Construct an empty dataframe replacing the game variables
    cols = ['game_id', 'patient_id', 'start_time',
            'session_id', 'active_duration', 'passive_duration', 'duration',
            'rom_min', 'rom_max', 'force_flexion','force_extension',
            'num_movements', 'game_rom_min', 'game_rom_max', 'game',
            'game_pressure_min', 'game_pressure_max', 'game_pressure_mean']

    df = pd.DataFrame(columns=cols)

    for idx, row in data.iterrows():
        for key in game_dict:
            try:
                dict = {
                    'game_id': game_dict[key],
                    }
'patient_id': row['patient_id'],
'start_time': row['start_time'],
'session_id': row['session_id'],
'active_duration': row['active_duration'],
'passive_duration': row['passive_duration'],
'duration': row['duration'],
'rom_min': row['rom_min'],
'rom_max': row['rom_max'],
'force_flexion': row['force_flexion'],
'force_extension': row['force_extension'],
'num_movements': row['num_movements'],
'game_rom_min': row[key + '_rom_min'],
'game_rom_max': row[key + '_rom_max'],
'game': row[key],
'game_pressure_min': row[key + '_pressure_min'],
'game_pressure_max': row[key + '_pressure_max'],
'game_pressure_mean': row[key + '_pressure_mean'],
}

df = df.append(dict, ignore_index=True)

except:
pass

return df

def concat_tall_dataframes(df_old, df_new):

    chunks = [df_old, df_new]
    return pd.concat(chunks, ignore_index=True)

def main():

    # I want to add some sort of a list to loop through

    for _ in range(1):

        file_list = [
            'user_session_data_with_pressure.txt',
            'user_session_data_with_pressure_2.txt',
            'user_session_data_with_pressure_dataframe_new.txt',
            'labelled_session_data.txt',]

        path_new = './data/raw_data/' + file_list[_]
data_wide = pd.read_csv(path_new, delimiter='|',
                        index_col=0)

game_dict = {
    'Console': 0,
    'Thermometer': 1,
    'Strongman': 2,
    'Strongman_Up': 3,
    'Space_Shooter': 4,
    'Blocks': 5,
    'Golf': 6,
    'Balloon_Rider': 7,
    'Brick_Breaker': 8,
    'Assessment': 26,
    'Cosmic_Tennis': 10,
    'Slot_Machine': 11,
    'Classic_Assessment': 12,
    'Coin_Push': 13,
    'Fishing': 14,
    'Symphony_Hero': 15,
    'Solitaire': 16,
    'Main_Menu': 17,
'Sphero': 18,
'Shape-Shooter': 19,
'Pegglung': 20,
'Plinko': 21,
'Countdown': 22,
'Tower': 23,
'Motus_Chef': 24,
'Motus_Pets': 25,
'Black_Jack': 27,
'Motus_Karts': 28,
'Candy_Swap': 29,
'Nick_Assessment': 30,
}

#df_new = tall_df(data_wide, game_dict)

#path_old = './data/Tall_DataFrame_Pressure.txt'
#df_old = pd.read_csv(path_old, delimiter='|', index_col=0)

#df_final = concat_tall_dataframes(df_old, df_new)
df = tall_df(data_wide, game_dict)

# Save file into the unlabeled processed data
df.to_csv(f'./data/processed_data/tall_unlabeled/Tall_DataFrame_Pressure{str(1)}.txt', sep='|')

if __name__ == '__main__':
    main()

B max_score_per_session.py

def create_unlabel_dataframe():

    #tall_dataframe_path = "./data/processed_data/tall_unlabeled/Tall_DataFrame_Pressure.txt"
    
    unlabeled_data = pd.read_csv(tall_dataframe_path,
        delimiter="|", index_col=0)
    unlabeled_data = unlabeled_data.drop(
        ["active_duration", "passive_duration"],
from min, max, axis=1

return unlabeled_data

def create_label_dataframe():
    label_path = "./data/raw_data/label_data/ROM Labelled Data - Sheet.csv"
    drop_label = ["Date of Assessment", "Max", "Min", "Assistance", "SW Version", "ROM Assessment Type",
                  "ROM Assessment Type"]

    return (pd.read_csv(label_path) .rename(columns={"ID": "patient_id"}) .drop(drop_label, axis=1))

def create_peripheral_dataframe():
    peripath = "./data/raw_data/label_data/"
Peripheral Type Lookup – Sheet.csv

```python
return pd.read_csv(peripath)

def create_score_dataframe():

    scorepath = "./data/raw_data/user_score_data.txt"
    score_data = pd.read_csv(scorepath,
        delimiter="|",
        index_col=0).rename(
            columns={"Unnamed: 0": "time_stamp"})

    score_drop = ["device_id", "dynamic_rom_max", "dynamic_rom_min",
                  "game", "session_index", "isTesting",
                  "is_low_rom", "is_no_rom", "pt_pressure_override",
                  "set_initial_rom"]

    score_data = score_data.drop(score_drop, axis=1)

    sort_score_columns = ["session_id",
                           "game_id",
                           "patient_id",
                           "score_id"]

    score_data = score_data.sort_values(sort_score_columns)
```
def create_max_score_dataframe():

    score_data = create_score_data()

    agg_func_max_score = {
        "score": "max",
    }

    # merge the df_max_game_score with the score_data to have
    # the "level" and "game_time" variables
```python
def create_max_score_game_dataframe():
    score_data = create_score_dataframe()
    max_game_score = create_max_score_dataframe()

    df_max_game_score = max_game_score.merge(score_data,
                                              on=["patient_id", "game_id", "session_id", "score"])

    df_max_game_score = df_max_game_score[['session_id', 'game_id',
                                           'score']]
```

def create_unlabeled_score():
    df_unlabeled = create_unlabeled_dataframe()
    df_score = create_score_dataframe()
    ls = ['patient_id', 'game_id', 'session_id']
    df_unlabeled_score = df_unlabeled.merge(df_score, on=ls)
    return df_unlabeled_score

def create_unlabeled_max_score():
    df_unlabeled = create_unlabeled_dataframe()
    df_max_score = create_max_score_game_dataframe()
    df_max_score = create_max_score_game_dataframe()
    df_max_score = create_max_score_game_dataframe()
    return df_unlabeled.merge(}
```python
def create_labeled_score():
    df_unlabeled_score = create_unlabeled_score()
    df_peripheral = create_peripheral_dataframe()
    labeled_data = create_label_dataframe()
    df_label_score = df_unlabeled_score.merge(labeled_data, on=['patient_id', 'game_id', 'session_id'])
    return df_label_score.merge(df_peripheral, on='patient_id').reset_index()

def label_max_score():
    df = create_labeled_score()
    return df.groupby(['session_id', 'patient_id']).agg({'score': 'max'}).reset_index()
```

def create_labeled_max_score():

    df_unlabeled_max_score = create_unlabeled_max_score()
    df_peripheral = create_peripheral_dataframe()
    labeled_data = create_label_dataframe()

    # merging labeled data with the unlabeled score data
    df_label_max_score = df_unlabeled_max_score.merge(labeled_data, on=["patient_id"])  
    return df_label_max_score.merge(df_peripheral, on="patient_id")
    df = df.drop("Peripheral Type", axis=1)

def max_score_per_session_game():

    df = create_labeled_max_score()

    df["start_time"] = pd.to_datetime(df["start_time"])
```python
def.sort_values(by="start_time")

# variable constraints like having negative pressure
df = cap_data(df)

score_col_sort = ["game_id", "patient_id", "session_id", "start_time",
                  "force_flexion", "force_extension", "num_movements",
                  "game_rom_min", "game_rom_max", "game",
                  "game_pressure_min", "game_pressure_max",
                  "game_pressure_mean",
                  "score", "peripheral_type",
                  "Classification", ]

df = df[score_col_sort]

# I am going to try to fill this missing data
# df = df.dropna()

# number of dropped duplicate values
df = df.drop_duplicates()
```
def fill_missing_data(df):
    def label_encoding(df):
        LE = LabelEncoder()
        # foot: 0, hand: 1
        df["peripheral_type"] = LE.fit_transform(df["peripheral_type"])
        # low: 0, high: 1, No: 2
        df["Classification"] = LE.fit_transform(df["Classification"])
        return df

        def label_encoding(df):
            LE = LabelEncoder()
            # foot: 0, hand: 1
            df["peripheral_type"] = LE.fit_transform(df["peripheral_type"])
            # low: 0, high: 1, No: 2
            df["Classification"] = LE.fit_transform(df["Classification"])
            return df

        def fill_missing_data(df):
            ""
            Missing data: ["game_rom_min", "game_rom_max", "game"]
            fillna(0) for ["game_rom_min", "game_rom_max"]
            ""

            missing_data = {"game_rom_min": df['game_rom_min'].mean(),
"game_rom_max": df['game_rom_max'].mean(),

"game": df['game'].min()}

return df.fillna(missing_data, inplace=True)

df = max_score_per_session_game()
REFERENCES


